Design and implementation of physical layer bitrate selection algorithms towards improving quality of service

By
Tzevelekidis Konstantinos

Dissertation for the degree of Master

Supervisor: Korakis Athanasios
Assistant professor

Tassiulas Leandros
Professor

Argyriou Antonios
Assistant professor

Volos, 2016

Πανεπιστήμιο Θεσσαλίας

Τμήμα ηλεκτρολόγων μηχανικών και μηχανικών υπολογιστών

Σχεδιασμός και υλοποίηση αλγορίθμων επιλογής υθμού μετάδοσης σε φυσικό επίπεδο προς επίτευξη βελτιωμένης ποιότητας υπηρεσίας

Τζεβελεκίδης Κωνσταντίνος

Μεταπτυχιακή Διατριβή

Επιβλέποντες καθηγητές : Κοράκης Αθανάσιος
Επίκουρος καθηγητής ΠΘ

Τασιούλας Λέανδρος
Καθηγητής ΠΘ

Αργυρίου Αντώνιος
Επίκουρος καθηγητής ΠΘ

Βόλος, 2016

Στην οικογένεια και στους φίλους μου,

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία αποτελεί την ολοκλήρωση των μεταπτυχιακών σπουδών μου στο τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας. Αποτελεί προϊόν εκτενούς αναζήτησης και έρευνας και πραγματοποιήθηκε με την υποστήριξη και καθοδήγηση συγκεκριμένων ανθρώπων, τους οποίους θα ήθελα να ευχαριστήσω θερμά.

Αρχικά, θα ήθελα να ευχαριστήσω τους επιβλέποντες καθηγητές μου, Αθανάσιο Κοράκη, Λέανδρο Τασιούλα και Αντώνιο Αργυρίου για την εμπιστοσύνη που μου έδειξαν, αναθέτοντάς μου τη συγκεκριμένη εργασία.

Θα ήθελα να ευχαριστήσω, ακόμη, το φίλο μου Dr. Στράτο Κερανίδη για την καθοδήγηση και την ουσιαστική συμβολή του στην περάτωση της εργασίας. Επίσης, θα ήθελα να ευχαριστήσω ιδιαίτερα τον καλό μου φίλο και συνάδελφο Ηλία Συρίγο, καθώς διαδραμάτισε σημαντικό ρόλο στην εξέλιξη της μεταπτυχιακής διατριβής προσφέροντας απλόχερα τη βοήθεια του οποτεδήποτε και αν χρειάστηκε, όπως και τους καλούς φίλους και συναδέλφους, Βιργίλιο Πασσά και Κωνσταντίνο Χούνο, για την υποστήριξή τους σε κάθε βήμα και τις επισημάνσεις τους στην εξέλιξη της εργασίας.

Τέλος, οφείλω να ευχαριστήσω την οικογένεια και τους φίλους μου για την υποστήριξη και τη βοήθειά τους, καθώς βρίσκονταν πάντα δίπλα μου σε όλη τη διάρκεια των σπουδών μου και με βοήθησαν να ανταπεξέλθω και να τις φέρω σε πέρας.

Τζεβελεκίδης Κωνσταντίνος

Βόλος, 2016

Περιληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η βελτιστοποίηση του αλγορίθμου προσαρμογής ρυθμού Minstrel, προκειμένου να επιτευχθεί μεγαλύτερος ρυθμός μετάδοσης (throughput) και καλύτερη παράδοση πακέτων (packet delivery rate), όταν ο σηματοθορυβικός λόγος (SNR) του καναλιού είναι μικρός. Υπάρχει πλήθος αλγορίθμων προσαρμογής ρυθμού για το επίπεδο ζεύξης (MAC) στο πρωτόκολλο IEEE 802.11, όπως είναι ο AMRR και ο Onoe. Ένας από τους πιο αποδοτικούς αλγορίθμους είναι ο Minstrel, ο οποίος διατηρεί στατιστικά για όλους τους ρυθμούς, βάση των οποίων επιλέγει και τον ρυθμό στον οποίο θα μεταδώσει τα πακέτα. Επίσης, οι αλλαγές στην ποιότητα του καναλιού, επηρεάζουν σημαντικά τις επιλογές του. Μέσα από πειράματα παρατηρήθηκε ότι σε διάφορες τοπολογίες, ο ρυθμός μετάδοσης και η παράδοση των πακέτων αυξάνεται όταν χρησιμοποιείται μεταβλητό μήκος πακέτου. Για μεταβλητό μήκος πακέτου υπάρχουν περισσότερες επιλογές για την καταλληλότερη επιλογή ρυθμού μετάδοσης με βάση την τωρινή κατάσταση του καναλιού. Προκειμένου να υποστηρίξει την παραπάνω λειτουργικότητα, ο αλγόριθμος Minstrel τροποποιήθηκε ανάλογα.

Λέξεις-κλειδιά: ασύρματες κάρτες δικτύου, προγραμματισμός, C, IEEE 802.11, πολλαπλή είσοδος πολλαπλή έξοδος (MIMO), συσσωμάτωση πλαισίων (frame aggregation), αλγόριθμοι προσαρμογής ρυθμού, Minstrel

ABSTRACT

The purpose of this thesis is to improve the Minstrel rate adaptation algorithm in order to achieve better performance in terms of throughput and packet delivery rate (PDR), when the channel condition deteriorates. There are many rate adaptation algorithms for MAC layer in IEEE 802.11 such as AMRR and Onoe. One of the most efficient rate adaptation algorithms is Minstrel, which is based explicitly on measured performance. It gathers statistics such as throughput and probability for all rates that have been used and based on that, selects the rate that is going to use in order to transmit the packets. Also, environmental changes affect significantly the choice of the rate. It was observed that throughput and jitter are enhanced when varying packet length is used. By varying the packet length more options are introduced for choosing the optimal configuration (rate and MTU) for the existing link quality. In order to support the aforementioned functionality, Minstrel rate adaptation algorithm was modified.

Keywords: wireless network cards, atheros, programming, C, IEEE 802.11, MIMO, frame aggregation, AMPDU, rate control algorithms, MinstrelS

THESIS SUBJECT

This thesis and the software developed in the context of, targets to contribute in the section of wireless networks. This software will be used in order to improve the quality of experience of IEEE 802.11 users.

The elaboration of this thesis was conducted with the help of Center of Research and Technology Hellas «CERTH», which has to present significant studies in the telecommunications sector.

TEXT STRUCTURE

The current dissertation is organized in four chapters, where each one describe

The **1st chapter** introduces the IEEE 802.11 standard. It presents the physical and the medium access control layers of the IEEE 802.11 standard.

The **2nd chapter** includes a detailed description of the PHY and MAC layer enhancements of IEEE 802.11n standard such as MIMO in PHY layer and frame aggregation in MAC layer.

In the **3rd chapter** there is a demonstration of various rate adaptation algorithms such as AMRR and SampleRate. A detailed explanation of Minstrel algorithm and of the incentives that led to this study follows. Subsequently, there is a description of the amendements implemented in the minstrel rate control algorithm. Finally, there is a detailed presentation of the experiments performed to show the improvements in throughput and packet data rate with the new implementation.

Finally in the **4th chapter,** there is the outlining of the objectives that were achieved and also are listed proposals for future enhancements.

LIST OF FIGURES

CHAPTER 1:

CHAPTER 2:

CHAPTER 3:

TABLE OF CONTENTS

1. INTRODUCTION

1.1 IEEE 802.11

Nowadays, wireless and cellular devices have a huge impact in our lives, as indicated by the rapidly increasing number of smart phones, personal and tablet computers in use (**Figure 1-1**). The most popular network in the wireless domain is the IEEE 802.11 wireless local area network (WLAN). Its advantages, such as interoperability, mobility, flexibility and cost effective deployment, have resulted in its wide distribution across enterprises, public sectors and homes.

During the last decade, however, the increasing demand for high data rate multimedia services over wireless, such as video streaming, voice over IP (VoIP), file transfer and online gaming, as well as the low efficiency of its medium access control (MAC) and physical (PHY) layer protocols, have become a barrier in its usage. In order to surpass these difficulties, the High-Throughput Study Group (HTSG) has tried to achieve higher data rates with the existing MAC and PHY mechanisms, which later was proved infeasible. Instead, they decided to introduce IEEE 802.11n, which was a high throughput (HT) extension of the current WLAN standard that would increase transmission rate and reduce compulsory overhead. The amendments of the new standard were the following:

- Use of frame aggregation in the MAC mechanism.
- Use of multiple input - multiple output (MIMO) antennas with orthogonal frequency division multiplexing (OFDM) and various channel bindings schemes in the PHY.

The network throughput in correlation with the previous standards (IEEE 802.11a and IEEE 802.11g) was significantly increased from 54Mbps to 600Mbps.

Figure 1-1: Internet of things

1.2 MAC and PHY Layers of IEEE 802.11

IEEE 802.11 is a set of MAC and physical layer specifications for implementing WLAN communications in various frequencies developed by the Institute of Electrical and Electronics Engineers (IEEE). The original specification of the IEEE 802.11 standard included a primitive MAC architecture and three basic over the air communication techniques, achieving maximal data rates of 1 and 2 Mbps. Given that these data rates were considerably low, various enhancements of IEEE 802.11 were deployed. These versions of IEEE 802.11 such as IEEE 802.11 a/b/g/ac/n use different frequency bands and offer various data rates (**Figure 1-2** and **Figure 1-3**). For example, IEEE 802.11a and IEEE 802.11g support data rates up to 54Mbps but operate in different frequency bands (2.4 and 5 GHz, respectively). Another example is IEEE 802.11n, which uses multiple antennas (MIMO), operating in both 2.4 and 5GHz and supporting data rates up to 600Mbps. The goal of these amendments was to improve PHY specifications (modulation and coding schemes), in order to attain higher data rates (**Figure 1-4**), nevertheless, in many cases it seems that the throughput measured at the

MAC layer was lower than expected. As a result, MAC seemed to be the barrier for achieving higher data rates.

Standard	Freq	Supported Rate	Number of Channels	Published	Market Introduction
802.11	2.4GHz	1, 2	11	1997	N/A
802.11b	2.4GHz	1, 2, 5.5, 11	11	1999	1999
802.11a	5GHz	6, 9, 12, 18, 24, 36, 48, 54	12	1999	2002
802.11g	2.4GHz	1, 2, 5.5, 11, 6, 9, 12, 18, 24, 36, 48, 54	11	2003	2003
802.11n	2.4/5GHz	100Mbps above MAC SAP	N/A	Expected 2007	Pre-N 2005

Figure 1-2: General information for some of the 802.11 standards (based on lectures presented for the course "Advanced Topics in Networks", University of Thessaly)

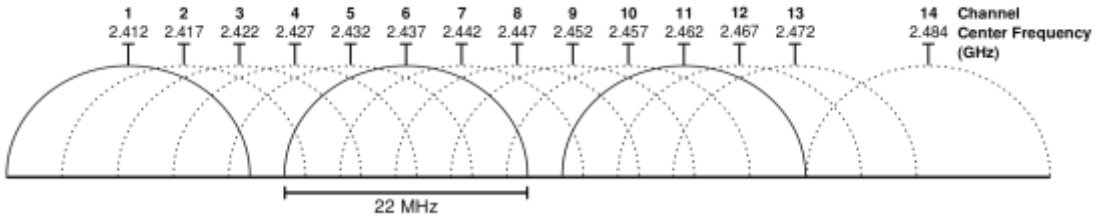


Figure 1-3: Available channels of IEEE 802.11 (from https://en.wikipedia.org/wiki/List_of_WLAN_channels)

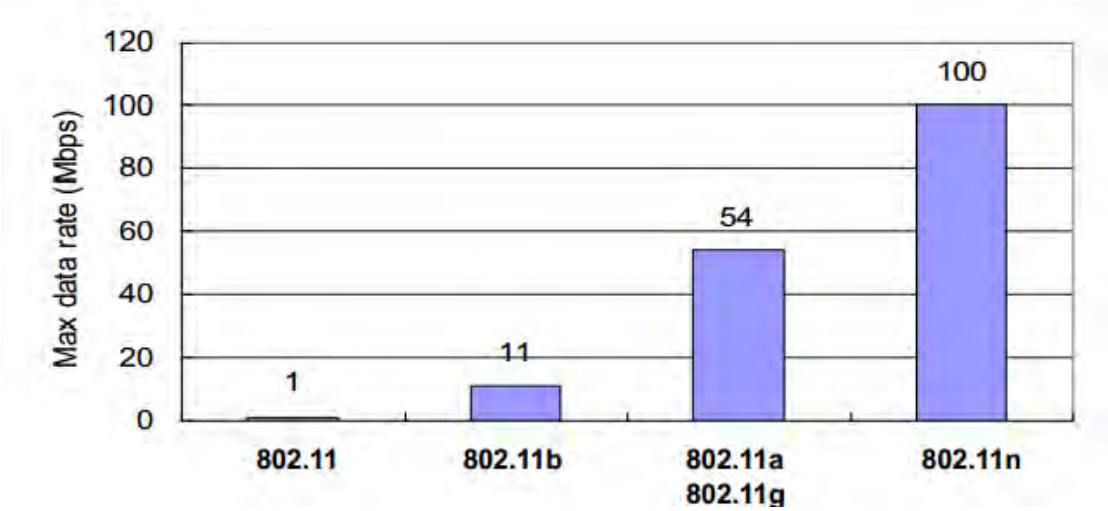


Figure 1-4: Max data rates for some of the 802.11 standards (based on lectures for the course "Advanced Topics in Networks", University of Thessaly)

MAC architecture supports fragmentation, encryption and acts as the interface between logical link control (LLC) sub layer and the network's PHY layer. In addition, it offers channel access control mechanisms, determining who and when will access the medium at any time. IEEE 802.11 uses two types of access schemes:

- Distributed Coordination Function (DCF).
- Point Coordination Function (PCF).

The first one is based on the carrier sense multiple access with collision avoidance (CSMA/CA) mechanism. Once the station is ready to send, starts sensing the channel (Clear Channel Assessment - CCA). If the channel is free for a short period of time (DCF Inter Frame Space - DIFS), then the station transmits. Otherwise, the station has to wait for a free DIFS, as well as for a random back off time selected within a contention window [0 – CW] (**Figure 1-5**). A random back off timer will begin to decrease as long as the medium is sensed idle for a DIFS. If a transmission is detected, the back off timer freezes and resumes once the channel is sensed as idle for a DIFS interval. When the random interval ends, the station senses the channel again and, in case it is idle, it sends the data frame through the wireless link. If the channel is still busy, the back off factor is set again and the process is repeated. The receiving station checks the frame for errors and in their absence, it acknowledges the reception of the packet after a short interframe space (SIFS). If the transmitting station does not receive the acknowledgment due to collision or transmission errors after a SIFS, it reschedules a new transmission. If the retransmission fails for a number of times the packet is dropped.

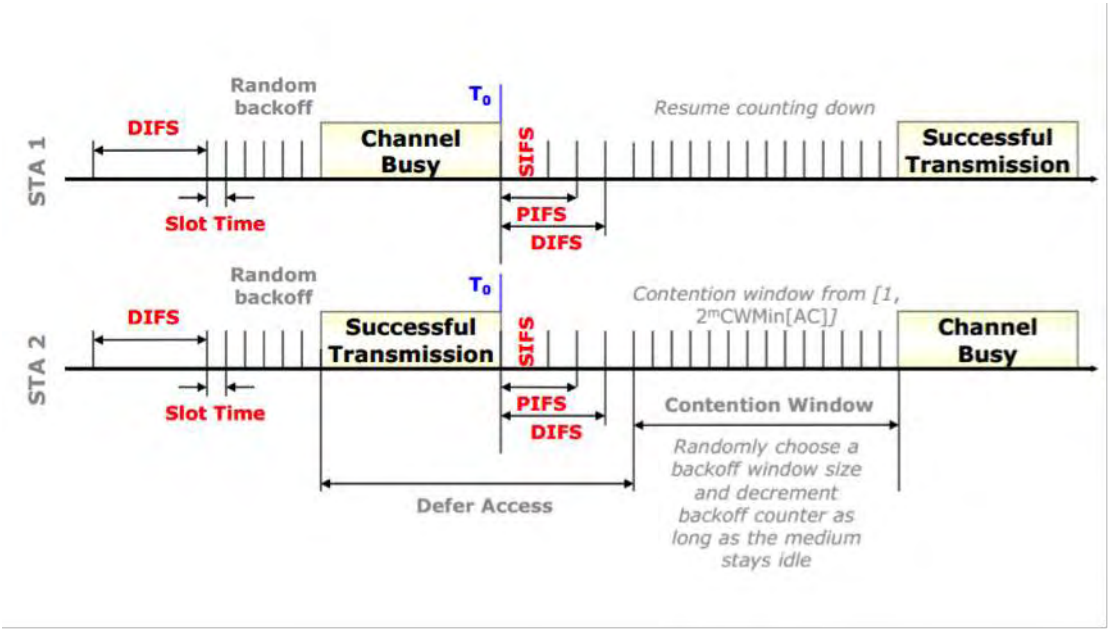


Figure 1-5: Basic DCF (based on lectures presented for the course "Telecommunication Networks", University of Thessaly)

In contrast to the carrier sense multiple access with collision detection (CSMA/CD) used by the Ethernet protocol (IEEE 802.3), the IEEE 802.11 does not use collision detection because:

- Collision detection acquires the station to be able to transmit and receive simultaneously, which is cost effective
- In case that one station has collision detection and fails to detect it during transmission, a collision can happen in the receiver (hidden terminal and fading problems).

In order to reduce frame collisions, a Request to Send / Clear to Send (RTS/CTS) mechanism was used in DCF. This mechanism addresses the problem of the hidden node terminal. Nevertheless, it is responsible for the exposed terminal problem in which a node postopones its transmission due to a neighbouring transmitter. This problem can be solved only if the nodes are synchronized and the packet sizes and data rates are the same for both transmitting nodes. When a node hears the RTS from a neighbour node but it does not receive the CTS, can deduce that the node is exposed and it is permitted to transmit to other neighbouring nodes.

RTS/CTS is a method to implement virtual carrier sensing. When a station wants to transmit a packet it has to send a RTS first. The receiving station will answer with a CTS. Other stations that hear the RTS-CTS indicate that the medium will be busy for the duration of the request and adjust the minimum time that must be elapsed before they can sense channel for idle status (Network Allocation Vector - NAV).

So, CSMA/CA (**Figure 1-6**) was implemented to avoid collisions rather than detecting and recovering from them.

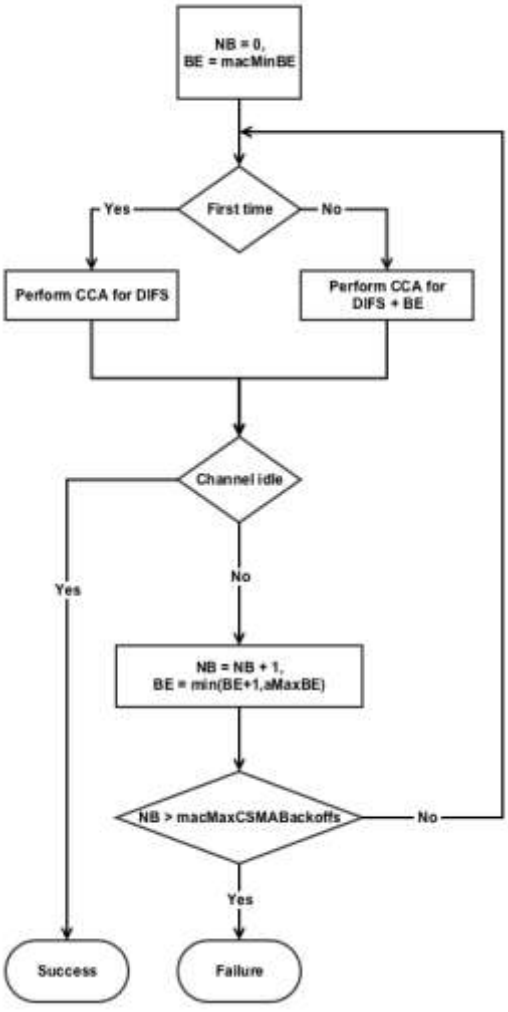


Figure 1-6: CSMA/CA mechanism

On the other hand, PCF is based on a poll and response mechanism. There is a point coordinator- access point (AP), which controls the communication within the network by polling stations according to a list. The AP waits for a period of time (PCF Interframe Space- PIFS) to occupy the channel. It sends a CF-Poll frame to the station capable to transmit a frame. If the coordinator has no available frames, it has to transmit null frame (**Figure 1-7**). PCF is a contention free mechanism and is adequate for delay sensitive applications.

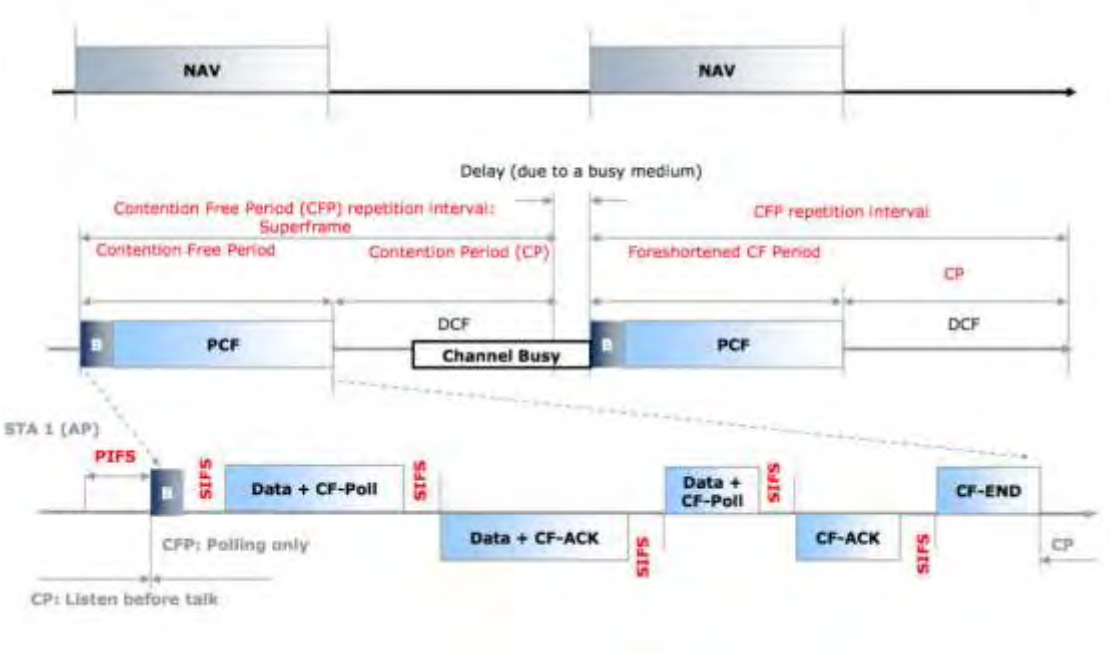


Figure 1-7: PCF (based on lectures presented for the course "Telecommunication Networks", University of Thessaly)

These schemes, however, were inadequate to resolve prioritization and differentiation between frames and multimedia applications. Hence, in order to provide a guaranteed level of quality of service IEEE 802.11e was introduced. IEEE 802.11e uses the Hybrid Coordination Function (HCF), which uses a contention based channel access method: the Enhanced DCF channel access (EDCA) where EDCA can operate simultaneously with a polling based HCF controlled channel access (HCCA). In order to improve MAC efficiency, IEEE 802.11e offers transmission opportunity (TXOP), an interval of time where multiple data frames can be transmitted from one station to the other.

Furthermore, IEEE 802.11e introduced block acknowledgment (Block ACK) where receivers can acknowledge multiple data frames, using an extended single ACK frame. At last, the QoS differentiation is achieved through the Traffic Identifiers (TID) and the Traffic Specification frames (TSPEC).

The additional overhead has major consequences in throughput in IEEE 802.11 when using DCF access scheme and poses the inefficiency of IEEE 802.11 over higher data rates. To be more precise, a complete transmission of a simple DCF consists of DIFS deferral, back off, data transmission, SIFS deferral, ACK transmission and propagation delay. It should be also mentioned that all physical layer headers and preambles would be transmitted in basic link rate, which is much less than the rate used for data transmission. In order to achieve a successful transmission, a significant overhead is accounted. As a result the higher the packet rate the higher the relative overhead the system introduces [1]. It is proved that the maximum ideal throughput is bounded at 50Mbps when the frame size is 1KB and the link rate increases to infinite [5]. When a legacy device uses a 54 Mbps link it can provide only a maximum relative MAC throughput that is less than 50% of the peak PHY (~25 Mbps) [1]. It is obvious that MAC and PHY layer overhead are critical and should be reduced to achieve higher data rates. In order to achieve this, aggregation can be used. Through aggregation several packets are grouped together and the overhead is calculated for the whole packet and not for every individual packet.

2. IEEE 802.11N

2.1 Physical Layer Enhancements

The IEEE 802.11n standard is an amendment to the previous IEEE 802.11 standards by adding multiple-input multiple-output (MIMO) and 40 MHz channels in the PHY layer and frame aggregation in the MAC sublayer. More specifically, MIMO offers antenna diversity in order to improve quality and reliability of the wireless link. It also supports spatial multiplexing in order to send independent signals (streams), which use the space dimension more than one time. Through using multiple antennas, an exploitation of the multipath phenomena can turn advantageous as the data throughput and range increases and the bit error decreases. In order to deal with the multipath nature of the channel, Orthogonal Frequency Devision Multiplexing (OFDM) coding is used along with the MIMO. The number of the OFDM data subcarriers is increased from 48 to 52 so higher throughput is achieved. Generally, MIMO increases the spectral efficiency of a wireless communication system in comparison with a single-input single-output (SISO). Also in multi antenna system, the space-time coding or channel state information available on the transmitter can be used to achieve diversity. Prior to transmission, in order to calibrate the radio channel, a session exchange of sounding PHY protocol data units (PPDUs) takes place. Based on that information, beam forming can be used to boost signal quality by selecting the proper power and coding scheme for the spatial streams. Higher signal quality means that a given data rate can be available at a longer range. For a given Signal to Noise Ration (SNR), a beam-formed transmission can carry higher data rate. Furthermore, different coding schemes are used for every transmission. MIMO has two coding schemes space-time block coding (STBC) and low-density parity check coding (LDPC). Also 5/6 coding rate is used instead of the 3/4 in the Forward Error Correction (FEC). Another difference is the shorter Guard Interval (GI), which was reduced from 800ns to 400ns. Finally, the usage of a wider channel (it was modified from 20 MHz to 40 MHz) improves the theoretical capacity limits.

In conclusion the PHY layer improvments in the IEEE 802.11n standard are:

- Modified OFDM
- Improved FEC
- Shorther GI
- Channel Bonding
- Spatial Multiplexing

If these mechanisms are employed in a network that contains only High Throughput (HT) nodes, the PHY rate can increase up to 600Mbps.

2.2 MAC Layer Enhancements

In 802.11n standard, there are also plenty of enhancements in the MAC layer such as bidirectional data transfer method over a single Transmission Opportunity (TXOP) (reverse direction), long network allocation vector (long-NAV) and frame aggregation, which is the most important MAC enhancement. All the previous enhancements lead to higher throughput and efficiency. Nonetheless, the amount of overhead of every frame, such as radio level headers, MAC frame fields, interframe spacing and acknowledgement of transmitted frames is significant in 802.11 devices and can consume a large part of the bandwidth. With frame aggregation this overhead can be decreased because several frames are combined into a larger frame. There are two methods of aggregation:

- Aggregate MAC Protocol Service Unit (A-MSDU)
- Aggregate MAC Protocol Data Unit (A-MPDU)

The main difference is that MSDU corresponds to the information that is imported to or exported from the upper part of the MAC sublayer from or to the higher layers, whereas MPDU relates to the information that the lower part of the MAC exchanges with the PHY layer. In order to make aggregation possible a protocol that acknowledges multiple MPDUs with a single block ACK in response to a block acknowledgement request (BAR) is used.

In particular, in **A-MSDU** multiple MSDUs for the same receiver first form a single MPDU and then are sent, enhancing MAC efficiency. In order to form the A-MSDU, a layer at the top of the MAC receives and buffers the MSDUs. An A-MSDU is completed either when the maximum number of MSDUs is reached or when the delay of the oldest packet reaches a predefined time limit. Its maximum length can be either 3839 or 7935 bytes, which can be found in the HT capabilities element advertised from an HT STA, in order to declare its HT status. Also the maximum delay can be set to an independent value but is usually set to 1μs in all stations. In order to form an MSDU, some requirements must be met:

1. All MSDUs have the same traffic identifiers (TID), which are used to characterize the traffic flows.
2. Lifetime of the A-MSDU should be analogous to the maximum lifetime of its MSDUs.
3. Destination address (DA) and sender address (SA) in the subframe header must match to the same receiver and transmitter address in the MAC header.

In the following figure (**Figure 2-1**), a simple structure of an A-MSDU is presented. Every subframe consists of a subframe header followed by the packet that arrived from the LLC and 0-3 bytes of padding, which should be multiple of four bytes except for the last one, so that the end receiver can identify the beginning of the next subframe. The A-MSDU lacks in error prone channels because if any subframe of the compressed MSDUs is corrupted, the whole A-MSDU should be retransmitted.

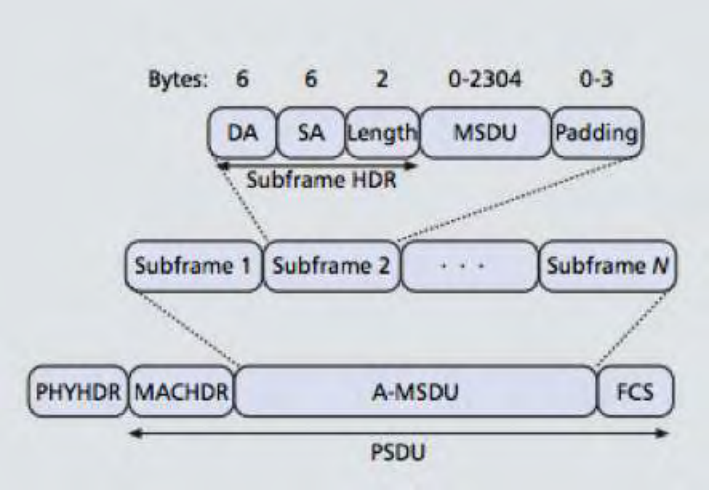


Figure 2-1: A-MSDU frame aggregation

In **A-MPDU,** multiple MPDUs are concatenated in one frame with a single PHY header. The main difference with A-MSDU is that A-MPDU takes place after the MAC header encapsulation. All the MPDUs within an A-MPDU should have the same receiver address but they do not need to have the same TID. Furthermore, the number of MPDUs used to form an A-MPDU depends on the number of packets already in the transmission queue (utmost 64 subframes) and the maximum length that can reach is 65535 bytes. In **Figure 2-2**, a depiction of the A-MPDU is presented. There is a set of fields -called MPDU delimiters and padding bytes, which define the structure of the A-MPDU. The former are inserted before each MPDU, in order to determine the MPDU position and its length inside the A-MPDU frame. In order to check the authenticity of the MPDU delimiter, a Cyclic Redundancy Check (CRC) is included in the delimiter. The padding bytes are added in the end, so that each MPDU is a multiple of four bytes in length. After the A-MPDU is received, de-aggregation takes place. For every subframe, MPDU delimiter is checked for errors using the CRC value and if none are found, it continues with the next subframe until it reaches the end of the A-MPDU. If it detects an error, it checks every 4 bytes until it locates a valid delimiter or the end of the A-MPDU. A-MPDU is less efficient than A-MSDU but it may perform better in high error rate environments, since it uses a mechanism called block-acknowledgment. With this mechanism, if one or more MPDU delimiters are received with errors, the A-MPDU can be recovered, as opposed to A-MSDU, where the whole frame should be retransmitted if one of the MSDU

contains bit errors.

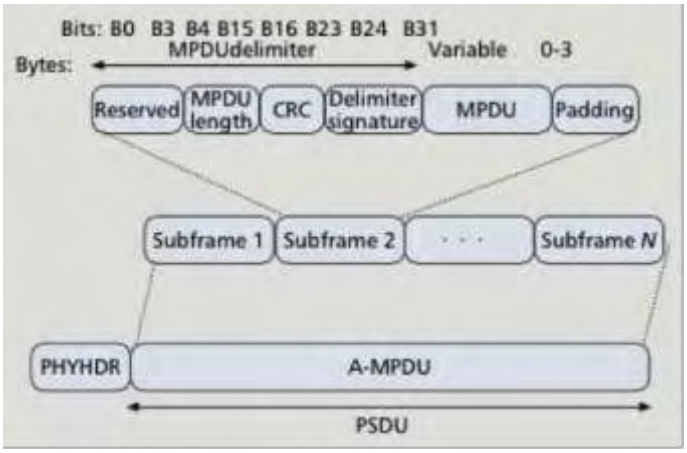


Figure 2-2: A-MPDU frame aggregation

Based on simulations, a comparison between the two models of aggregation is made (Dionysios Skordoulis et al. 2008). The testing environment included a fixed HT AP and a fixed HT STA with 2 antennas and the distance between them was ten meters with Line Of Sight (LOS). They were both operating in a 20-MHz channel, using for Modulation and Coding Scheme (MCS) the 64-QAM ¾. The AP provided loads of User Data Protocol (UDP) traffic without timeout values and with the same TIDs for 10 seconds. Some assumptions were made such as that the transmission proceeded with no interference or channel fading and that all frames were received successfully. **Figure 2-3**, depicts the comparison between MAC throughput and the offered load (OL) for 3 different settings of frame aggregation (A-MSDU, A-MPDU, no aggregation) where the OL is increased by just varying the packet size while keeping constant packet rate (CPR) to 40μs. In the first simulation the OL begins from 25 Mbps and increments up to 300 Mbps with the increase of the packet size (125/250/500/750/1000/1500 bytes). In addition, the traffic generation rate is configured high, in order to saturate the air link rate (~144 Mbps) and the maximum A-MSDU length is 4KB. In both aggregation types, throughput increases. It is obvious that for smaller packet sizes, all types of aggregation have the same result. When the packet size becomes larger than 250 bytes, A-MPDU operates much better than A-MSDU, as fewer MSDUs fit in an A-MSDU. Furthermore, throughput of A-MSDU drops when the packet size is bigger than 1000 bytes because the A-MSDU fits less than 4

MSDUs. If aggregation is not used, throughput always increases as OL increases, however the achieved throughput is 3 times lower than the throughput of the A-MPDU aggregation.

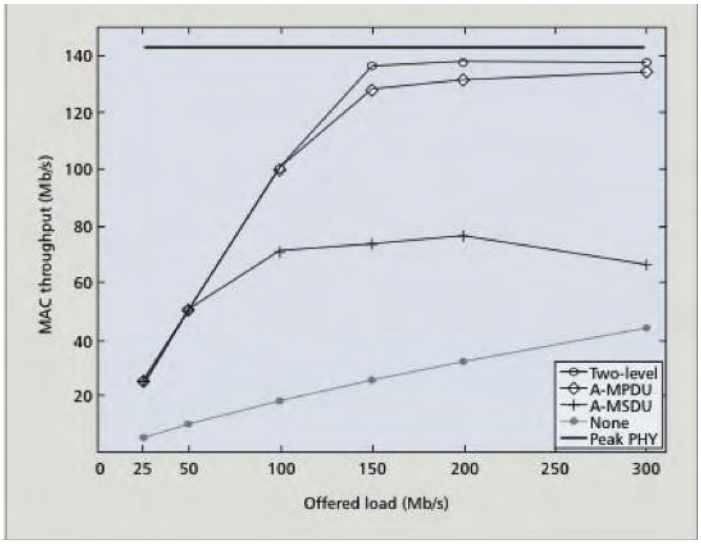


Figure 2-3: Throughput vs Increased offered load by varying packet size

In the second simulation, the OL increases by altering the variable packet rate (VPR) and the packet size is constant to 1000 bytes. In **Figure 2-4**, the behavior of throughput is similar to the previous simulation. At the beginning, throughput increases as OL increases and A-MPDU operates better than A-MSDU. When the channel is saturated, however, the throughput of all channels remains constant, which is characterized as normal because when the channel is saturated the demand cannot be met.

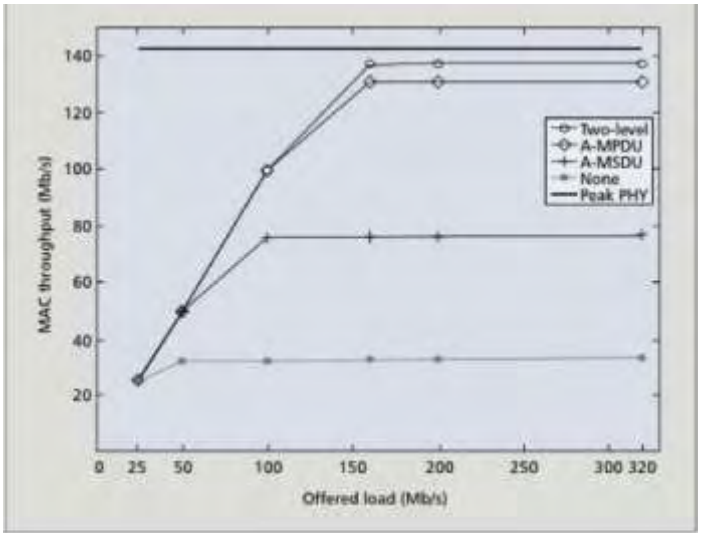


Figure 2-4: Throughput vs Increased offered load by varying the packet arrival interval using constant packet size = 1kB

As a result of the simulations, the A-MSDU does not utilize the channel as well as A-MPDU. However, in an ideal environment both types of aggregation minimize the PHY/MAC overhead and perform better than the legacy 802.11 standards but in a real working environment, there are still many issues to resolve, such as the complexity of the aggregation that can lead to larger delays.

A disadvantage of frame aggregation is that for a larger aggregated frame more processing time is needed, hence the station has to wait longer before its next chance for channel access. Additionally, channel conditions are correlated with MAC efficiency. For example, if a large frame is corrupted in a channel with high error rate, it may waste a long period of channel time. In the case of A-MPDU aggregation where the frame is made of the buffered packets, if CPR is low, the efficiency will also be small.

Several studies investigate the effects of the packet size in an error prone channel for IEEE802.11 DCF, showing that, in order to achieve a maximum throughput, an optimal frame size should be picked under a certain bit error rate (BER) (Yin et al. 2004). Also, simulation results show that there are significant throughput improvements, when compared with the randomized and fixed frame aggregation algorithms (Yuxia Lin et al. 2006).

3. IEEE 802.11 RATE CONTROL ALGORITHMS

3.1 Rate adaptation schemes

Rate adaptation is a link layer mechanism which is critical for the performance of IEEE 802.11 based networks. The main idea is that the transmitter selects the transmission rate offering the maximum throughput and dynamically adapts its decision to the continuously varying channel conditions without having information from the receiver.

Many rate control algorithms (RCA) for MAC layer in IEEE 802.11 standard exist. Most of them use Received Signal Strength Indication (RSSI) as a criterion for choosing the proper rate. Nevertheless, this can be misleading, as it does not take into consideration the multipath effect. Furthermore, the possibility of a successful packet transmission is a combination of three aspects: the distance between the devices, the multipath phenomenon and the interference of the environment. In a wireless environment, these variables cannot be defined and are unpredictable, making the selection of the proper rate difficult.

There are two main approaches in the design of rate adaptation schemes for 802.11 wireless networks:

- Algorithms based on signal strength
- Algorithms based on statistics

In the former category, the rate adaptation algorithm relies on wireless signal measurements, such as Signal-to-Noise Ratio (SNR) or Received Signal Strength Indicator (RSSI) in order to choose the transmission rate with the best performance, usually the maximum link-layer throughput. In order to calculate the SNR, the sender transmits RTS frames to estimate it and the receiver sends back this information through the CTS frames. For the RSSI approach, RSSI value is directly measured on the sender. The main drawback of these algorithms is that they require an accurate channel model in order to correlate the signal strength measurements with the corresponding link performance, which may be impractical for unstable environments. Examples of these rates are Receiver

Based AutoRate (RBAR) and Opportunistic AutoRate (OAR) (Emilio Ancilloti et al. 2008).

In the second category, information such as number of retries, number of successfully transmitted packets and number of failures is used in order to pick the best rate. An example of this type of schemes is Automatic Rate Fallback(ARF). In ARF a probe packet is sent after either ten consecutive successfully transmitted packets or a timeout. The probe packet is transmitted in a bit rate higher than the one in use. Based on the success or failure of the transmission of the probing packet the rate adopts accordingly. Other examples of such rate control algorithms are Adaptive ARF (AARF), Adaptive Multi Rate Retry (AMRR) and ONOE. However, these types of rate adaptation schemes suffer from a number of drawbacks. First of all there is not always a correlation between frame loss and future channel conditions. Furthermore, the estimation for the channel quality might be inefficient when a small amount of probing packets is used and can be easily affected by medium contention.

In the next section some rate control algorithms will be presented.

3.1.1 AMRR

The AMRR exploits the capability of the MadWifi driver that allows the network interface to transmit the frame retransmissions at different data rates. It uses a mechanism called retry chain. Four rates (r_0, r_1, r_2, r_3) and four transmission counts (c_0, c_1, c_2, c_3) are associated with each frame. Rate r_0 is the first rate used in order to transmit the frame. If transmission fails for c_0 -1 retries, the r_1 rate is used for c_1 times and so on. If all the rates fail, the frame is discarded. In AMRR retry chain, the numbers of retransmissions (c_0, c_1, c_2, c_3) for every rate are equal to one, which means that every rate is tried once. Rate r_3 is the lowest base rate and r_0 is the best rate. The selection of the best rate is based on the number of transmission failures in the last period (default value is one second). If for more than 10 frame transmissions, the failures are less than 10% then a higher rate is chosen. Otherwise, if more than 33% of the transmissions failed, a lower rate is chosen (Emilio Ancilloti et al. 2008).

3.1.2 ONOE

ONOE is similar with AMRR algorithm. It uses also the retry chain but with more retransmissions for every rate (i.e. $c_0 = 4$, $c_1 = c_2 = c_3 = 2$). However, the main difference is that ONOE is a credit based rate control algorithm. Specifically, if for more than 10 frame transmissions in the last period, the failures are less than 10%, the credits are incremented by one. If not, the credit is deducted by one. If the total number of credits for a specific rate is more than a threshold, then the next available higher rate is chosen as r_0. If more than 50% of the transmissions fail in the last period, the current rate is decreased. If the transmission using a specific data rate has failed in the last period, Onoe will not attempt to select it, unless 10 seconds have elapsed since the last attempt. Whenever rate changes, the credits are resetted (Emilio Ancilloti et al. 2008).

3.1.3 SampleRate

SampleRate selects the transmission data rate, which has the smallest average packet transmission time as measured by recent samples. Every tenth packet, it chooses a random rate from a set of data rates that can perform better than the current one and uses this rate to send the packet in order to estimate the average transmission time. Data rates with several successive failures are excluded from the sampling procedure. In order to adapt to the volatile channel conditions, calculation of the average transmission time is done using packets that were sent during the last ten seconds. SampeRate stops using a rate if it faces four consecutive failures or if its lossless transmission time is greater than the average transmission time of the current rate. If in the sampling process, there aren't any received acknowledgments or if the number of packets sent is multiple of 10, SampleRate transmits the packet with the highest rate, which has not failed four successive times (J. C. Bicket 2005).

3.1.4 Performance Evaluation

Through experiments Sourav Pal et al, present the performance analysis of the aforementioned rate adaptation algorithms. Specifically, they presented how rate adaptation algorithms react to the variations of received signal strength indicator (RSSI). In particular, when RSSI is decreased, a lower data rate is chosen in all RCAs. However, each algorithm behaves differently when the link condition is low and when short-term link flunctuations exist. For example Onoe is insensitive to minor variations of RSSI in contrast to AMMR, which is the most sensitive one.

Sourav Pal et al, conducted also experiments with different traffic categories (VoIP, video streaming, web browsing and file download) in order to calculate the impact of the RCAs on application level throughput. As shown in **Figure 3-1**, they found that there are significant differences between uplink and downlink throughput. It was proved that AMRR performs more effective in non-real time traffic. SampleRate's performance is quite the same with AMRR's except the VoIP applications where the SampleRate is better.

Rate Control Algorithms	VoIP		Streaming	Interactive		Elastic
	Up	Down		Up	Down	
Onoe	73.89	69.076	936.48	20.05	91.37	1946.91
AMRR	36.47	42.24	1243.95	26.11	109.99	3295.87
SampleRate	50.49	66.92	1132.17	26.07	97.79	2617.32

Figure 3-1: Average throughput (Kbps) for different traffic classes for the aforementioned RCAs

So the state of the wireless channel is significant for the RCAs and packet probes and RSSI values seem to be insufficient for estimating the wireless channel condition. Furhtermore, the efficiency of the RCAs can be increased, if the type of the traffic (e.g. VoIP, data transfer) is taken into consideration.

3.2 Minstrel

Minstrel is a one of the most efficient rate control algorithms for the MAC layer. It was designed and implemented in MadWifi wlan driver for linux, by Derek Smithies Ph.D (it was ported by Felix Fietkau). Minstrel is the implementation of Sample Rate in linux. This algorithm was created based on the observation that with the existing rate control algorithm the chosen rate was not the optimal one. It also seemed that if there were some environmental changes the rate did not adapt correctly. Be that as it may, if all the above were taken into consideration, better throughput could be achieved. So minstrel is based explicitly on measured performance and on the environmental changes. In minstrel all rates are tried regularly, meaning that if a rate works well it is used, otherwise it is ignored.

As shown in **Figure 3-2**, minstrel has a rate table (content of rc_stats file) for gathering statistics for every rate. The statistic table consists of the name rate, throughput, EWMA probability, this prob, this succ/attempts, success, attempts, total packet count, the ideal and the look around packets. Every second, the table is evaluated 10 times. Specifically, left of the rate names there are the letters T, t and P, which indicate the rates with the highest throughput, second highest throughput and the highest EWMA probability. The throughput column describes the measured throughput for a packet. The EWMA probability is the current time weighted chance that a packet will reach the remote station. This prob reports the success chance from the last time interval in which minstrel records data. This succ/attempt indicates how many packets were sent and how many of them were sent successfully in the last time interval, whereas success and attempts represent the same values for the whole time of the association. At last, the total packet count (modulo 10000) represents the number of packets that have been processed by minstrel including also those sent for randomly trying other rates, which is usually 10% of the total number of packets sent.

rate	throughput	ewma prob	this prob	this succ/attempt	success	attempts
P1	0.9	99.9	100.0	0(0)	105	111
2	0.4	25.0	100.0	0(0)	1	1
5.5	1.2	25.0	100.0	0(0)	1	1
11	1.1	12.5	50.0	0(0)	1	2
6	0.0	0.0	0.0	0(0)	0	0
9	0.0	0.0	0.0	0(0)	0	0
12	0.0	0.0	0.0	0(0)	0	0
18	0.0	0.0	0.0	0(0)	0	0
24	0.0	0.0	0.0	0(0)	0	0
36	0.0	0.0	0.0	0(0)	0	0
t48	16.0	40.9	88.8	0(0)	9	10
T 54	16.2	91.1	91.2	115(126)	96429	109032

Total packet count:: ideal 5756 lookaround 641

Figure 3-2: Rate table (rc_stats file) for legacy rates

In order to measure the efficiency of the minstrel rate control algorithm, throughput was used. Throughput is defined as :

$$\frac{\Pr ob\_success\_transmission * Megabits\_transmitted}{time\_for\_1\_try\_of\_1\_packet\_to\_be\_sent\_on\_the\_air}$$

For every packet that is transmitted throughput is calculated and later used to decide which rate is the most optimal. In order to have a more accurate decision, however, an amount of packets should be sent in rates that are not considered as optimal. Minstrel algorithm uses 10% of the packets to randomly try other rates and calculate the statistics.

The decision for the best rate is made based on:

- Best throughput
- Second best throughput
- Highest probability of success

The above data are used to form the retry chain. For example Atheros IEEE 802.11 abg chipset has 4 segments. Each segment is an advertisement to the hardware to try to send the packet in a specific rate (r_0, r_1, r_2, r_3) for a particular number of tries (c_0, c_1, c_2, c_3). If the packet failed to be sent in the first rate (r_0) for c_0 times, the next rate r_1 is tried for c_1 times. The process continues until the packet is sent successfully. If all rates fail the packet will be dropped. In minstrel

the number of attempts was based on the desire to get the packet sent in less than 26ms. The retry chain is calculated differently if the packet is a normal transmission packet (90% of total packets) or if it is a sample packet (10% of total packets). In the former case the retry chain is best throughput, next best throughput, best probability and lowest base rate. In the latter case, the retry chain is random look around, best throughput, best probability and lowest base rate (**Figure 3-3**).

Try	Lookaround rate	Normal rate
1	Random Lookaround	Best Throughput
2	Best Throughput	Next Best Throughput
3	Best Probability	Best Probability
4	Lowest Baserate	Lowest Baserate

Figure 3-3: Retry chain format

Based on experimental work, Felix Fietkau showed that performance of this retry chain is not so optimal. The time spent on sampling should be more adaptive, depending on the condition of the link. If the link is good, the time spent on sampling lower rates should be less than when the link condition deteriorates. In order to determine that the slower rates will be sampled, the retry chain was modified as shown in **Figure 3-4**. In this retry chain, the random rate is placed in the second place if it is lower than the best throughput rate, otherwise it is placed in the first position. Thus, if the best throughput is 54 Mbps the slower rates are sampled when the packet fails in transmission.

Try	Lookaround rate		Normal rate
	random < best	**random > best**	
1	Best Throughput	Random Rate	Best Throughput
2	Random Rate	Best Throughput	Next Best Throughput
3	Best Probability	Best Probability	Best Probability
4	Lowest Baserate	Lowest Baserate	Lowest Baserate

Figure 3-4: Modified retry chain

In order to measure the success history of every rate, Minstrel algorithm uses Exponential Weighted Moving Average (EWMA) calculations. With the use of EWMA recent results have bigger impact in the rate selection from the older. The EWMA calculation is carried out 10 times per second for every rate, except for the case where there are no packets for a particular rate in a time interval. The update interval is 100 ms and this value was chosen, in order to minimize the amount of CPU time spent for the updates, to have a quick recovery in case of a bad rate selection and to manage to collect enough information about the successes and the failures.

3.3 Incentives

The critical factor in order to measure the performance of any communication system is the signal to noise ratio (SNR). Better performance is achieved when the SNR is higher. As SNR decreases, lower data rates are used. If SNR is lower than a specific threshold, communication will not be supported. Specifically, each data rate requires a minimum received signal power level for reliable operation. This minimum received signal power level is called receive sensitivity. If the received signal power level is less than the receive sensitivity for a data rate, this data rate can no longer be supported and a lower rate should be chosen. In **Figure 3-5** are listed the modulation and coding schemes for 802.11n.

MCS RATE INDEX	DATA STREAMS	MODULATION / ECC	DATA RATE (MBPS)				RECEIVE SENSITIVITY (DBM)	
			800NS GI		400NS GI			
			20 MHZ	40 MHZ	20 MHZ	40 MHZ	20 MHZ	40 MHZ
0	1	BPSK / 1:2	6.5	13.5	7.2	15.0	-82	-79
1	1	QPSK / 1:2	13.0	27.0	14.4	30.0	-79	-76
2	1	QPSK / 3:4	19.5	40.5	21.7	45.0	-77	-74
3	1	16-QAM / 1:2	26.0	54.0	28.9	60.0	-74	-71
4	1	16-QAM / 3:4	39.0	81.0	43.3	90.0	-70	-67
5	1	64-QAM / 2:3	52.0	108.0	57.8	120.0	-66	-63
6	1	64-QAM / 3:4	58.5	121.5	65.0	135.0	-65	-62
7	1	64-QAM / 5:6	65.0	135.0	72.2	150.0	-64	-61
8	2	BPSK / 1:2	13.0	27.0	14.4	30.0	-82	-79
9	2	QPSK / 1:2	26.0	54.0	28.9	60.0	-79	-76
10	2	QPSK / 3:4	39.0	81.0	43.3	90.0	-77	-74
11	2	16-QAM / 1:2	52.0	108.0	57.8	120.0	-74	-71
12	2	16-QAM / 3:4	78.0	162.0	86.7	180.0	-70	-67
13	2	64-QAM / 2:3	104.0	216.0	115.6	240.0	-66	-63
14	2	64-QAM / 3:4	117.0	243.0	130.0	270.0	-65	-62
15	2	64-QAM / 5:6	130.0	270.0	144.4	300.0	-64	-61
16	3	BPSK / 1:2	19.5	40.5	21.7	45.0	-82	-79
17	3	QPSK / 1:2	39.0	81.0	43.3	90.0	-79	-76
18	3	QPSK / 3:4	58.5	121.5	65.0	135.0	-77	-74
19	3	16-QAM / 1:2	78.0	162.0	86.7	180.0	-74	-71
20	3	16-QAM / 3:4	117.0	243.0	130.7	270.0	-70	-67
21	3	64-QAM / 2:3	156.0	324.0	173.3	360.0	-66	-63
22	3	64-QAM / 3:4	175.5	364.5	195.0	405.0	-65	-62
23	3	64-QAM / 5:6	195.0	405.0	216.7	450.0	-64	-61

Figure 3-5: 802.11n Association Data rates

Every data rate has its receive sensitivity threshold. The more close to the receive sensitivity the received signal power is, the better results are achieved. It is observed that the difference in receive sensitivity between two adjacent rates is bigger than other rates. For example the difference for rate 3 (-74dBm) and rate 4 (-70dBm) is 4 dBm but for rate 5 (-66 dBm) and rate 6 (-65 dBm) is 1 dBm. So, in cases such as the one with the rates 3 and 4, if the received power signal is between the receive sensitivities of the two rates (e.g. -72 dBm), rate 3 will be used but it won't be an optimal choice. However, if between rate 3 and rate 4 were additional configurations that cover the sensitivity thresholds between -74dBm and -70 dBm, there would be a better match.

It was observed, that for some network topologies, modifying the packet length led to higher throughput and less jitter. This happens for example by reducing the size of the packet and thus increasing the probability of the correct reception and decoding. And so, a rate that was not supported for the default packet size, now can be used for transmission.

This leads to different receive sensitivity thresholds for varying packet sizes, introducing more options for choosing the optimal configuration (rate and MTU) for the existing link quality.

3.4 Proof of concept

In order to show the above observations, a default Atheros driver with minstrel's rate adaptation algorithm enabled, was used in different network topologies. A lot of experiments with different types of topologies took place, in order to show how MTU affects PDR, throughput and jitter. All the experiments were done in the NITOS testbed and especially in Indoor RF Isolated testbed and in Outdoor testbed (**Figure 3-6**) and in EKETA's office testbed.

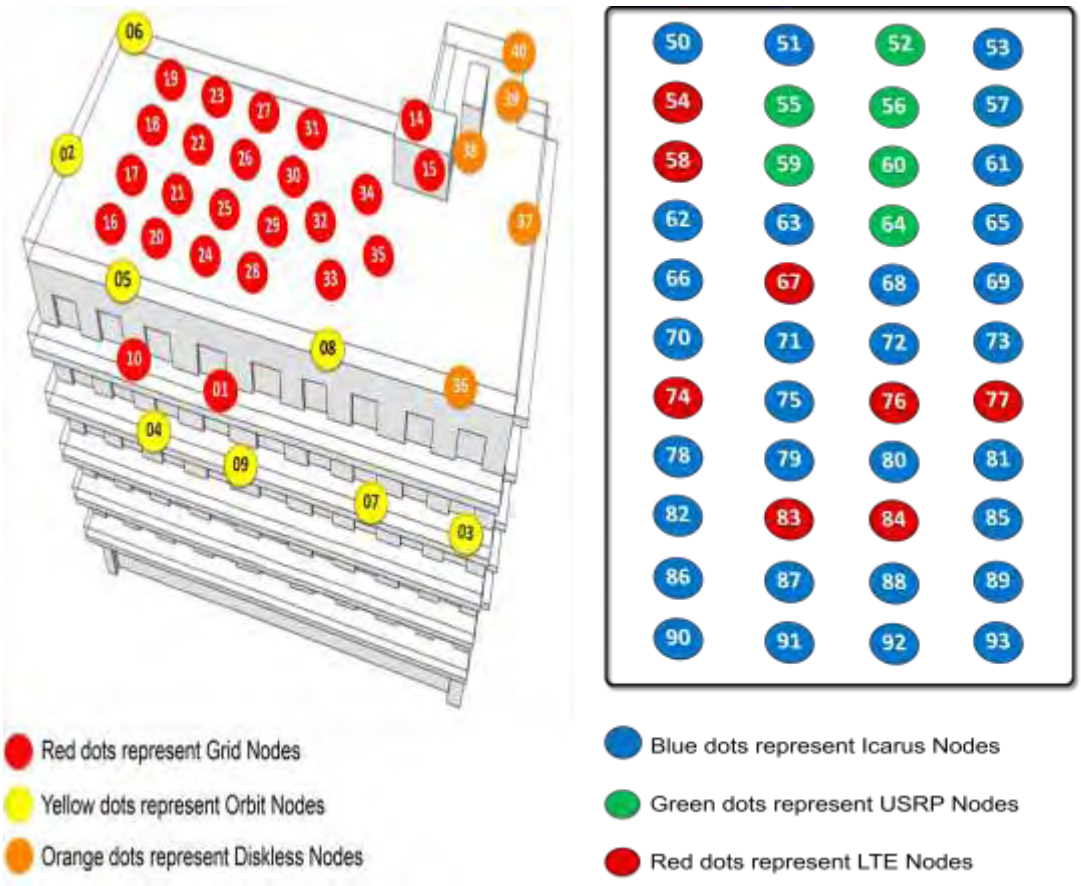


Figure 3-6: Nitos testbeds

Three scenarios were tested:

1. High SNR scenarios
2. Low SNR scenarios
3. Hidden terminal scenarios

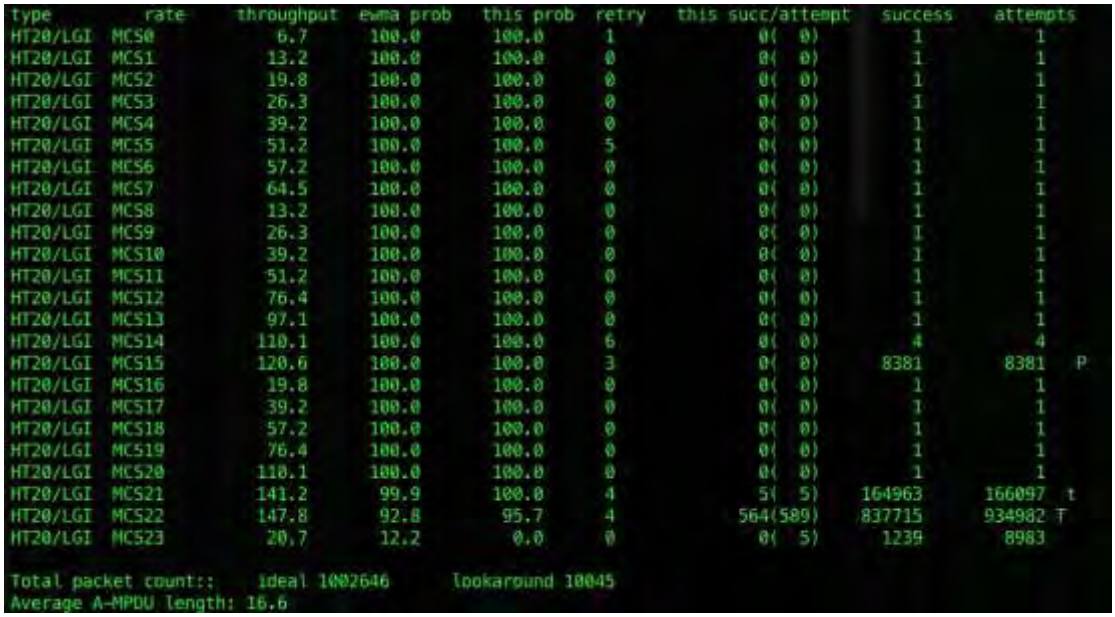
Different topologies were employed for all scenarios, in order to have more accurate results. More specifically, for the high SNR scenarios two adjacent nodes were chosen and for the low SNR the nodes were chosen to be as far as possible. In order to deteriorate the signal quality one antenna was used for transmitting (tx) and receiving (rx). Also, for both scenarios Request to Send/Clear to Send (RTS/CTS) mechanism was disabled. In high and in low SNR topologies, one of the node acts as an access point (AP) and the other node as an attached node (STA). In the hidden terminal scenario there was also a node that was sending traffic with a constant bit rate. In the primary experiments the default Atheros driver was used and in the following ones the modified driver. Traffic from the attached node to the AP is sent using iperf command (**Figure 3-7**, **Figure 3-8**). While the traffic is sent, the driver selects the MCSs with the best throughput (T), the second best throughput (t) and the best probability (P). When the driver has stabilized the decision about the aforementioned MCSs (**Figure 3-9**), the rate is fixed to the MCS with the best throughput (T). For a set of MTUs throughput, jitter and PDR are gathered.

```
root@node092:~# iperf -c 192.168.1.1 -u -i 1 -b 500M -t 1000
------------------------------------------------------------
Client connecting to 192.168.1.1, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  224 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.2 port 36039 connected with 192.168.1.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  17.1 MBytes   143 Mbits/sec
[  3]  1.0- 2.0 sec  14.5 MBytes   122 Mbits/sec
[  3]  2.0- 3.0 sec  14.7 MBytes   124 Mbits/sec
[  3]  3.0- 4.0 sec  14.5 MBytes   122 Mbits/sec
[  3]  4.0- 5.0 sec  14.5 MBytes   122 Mbits/sec
[  3]  5.0- 6.0 sec  14.6 MBytes   122 Mbits/sec
[  3]  6.0- 7.0 sec  14.7 MBytes   124 Mbits/sec
[  3]  7.0- 8.0 sec  14.5 MBytes   121 Mbits/sec
[  3]  8.0- 9.0 sec  14.4 MBytes   121 Mbits/sec
[  3]  9.0-10.0 sec  14.7 MBytes   123 Mbits/sec
[  3] 10.0-11.0 sec  14.3 MBytes   120 Mbits/sec
[  3] 11.0-12.0 sec  14.4 MBytes   121 Mbits/sec
[  3] 12.0-13.0 sec  14.7 MBytes   124 Mbits/sec
[  3] 13.0-14.0 sec  14.5 MBytes   122 Mbits/sec
[  3] 14.0-15.0 sec  14.5 MBytes   122 Mbits/sec
[  3] 15.0-16.0 sec  14.6 MBytes   123 Mbits/sec
[  3] 16.0-17.0 sec  14.7 MBytes   124 Mbits/sec
[  3] 17.0-18.0 sec  14.5 MBytes   121 Mbits/sec
[  3] 18.0-19.0 sec  14.7 MBytes   124 Mbits/sec
[  3] 19.0-20.0 sec  14.4 MBytes   120 Mbits/sec
[  3] 20.0-21.0 sec  14.5 MBytes   121 Mbits/sec
```

Figure 3-7: Iperf client side

```
root@node091:~# iperf -s -u -i 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  224 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.1 port 5001 connected with 192.168.1.2 port 36039
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0- 1.0 sec  14.0 MBytes   117 Mbits/sec   0.441 ms 2812/12774 (22%)
[  3]  1.0- 2.0 sec  14.3 MBytes   120 Mbits/sec   0.041 ms  233/10415 (2.2%)
[  3]  2.0- 3.0 sec  14.3 MBytes   120 Mbits/sec   0.060 ms  295/10479 (2.8%)
[  3]  3.0- 4.0 sec  14.2 MBytes   119 Mbits/sec   0.111 ms  216/10314 (2.1%)
[  3]  4.0- 5.0 sec  14.1 MBytes   118 Mbits/sec   0.061 ms  321/10390 (3.1%)
[  3]  5.0- 6.0 sec  14.2 MBytes   119 Mbits/sec   0.123 ms  236/10380 (2.3%)
[  3]  6.0- 7.0 sec  14.5 MBytes   121 Mbits/sec   0.098 ms  216/10524 (2.1%)
[  3]  7.0- 8.0 sec  14.1 MBytes   118 Mbits/sec   0.103 ms  264/10327 (2.6%)
[  3]  8.0- 9.0 sec  13.9 MBytes   117 Mbits/sec   0.068 ms  334/10248 (3.3%)
[  3]  9.0-10.0 sec  14.4 MBytes   121 Mbits/sec   0.089 ms  291/10539 (2.8%)
[  3] 10.0-11.0 sec  14.0 MBytes   118 Mbits/sec   0.131 ms  242/10264 (2.4%)
[  3] 11.0-12.0 sec  14.1 MBytes   118 Mbits/sec   0.119 ms  134/10199 (1.3%)
[  3] 12.0-13.0 sec  14.4 MBytes   121 Mbits/sec   0.066 ms  264/10562 (2.5%)
[  3] 13.0-14.0 sec  14.1 MBytes   118 Mbits/sec   0.149 ms  259/10295 (2.5%)
[  3] 14.0-15.0 sec  14.2 MBytes   119 Mbits/sec   0.058 ms  261/10413 (2.5%)
[  3] 15.0-16.0 sec  14.2 MBytes   119 Mbits/sec   0.061 ms  312/10433 (3%)
[  3] 16.0-17.0 sec  14.4 MBytes   120 Mbits/sec   0.119 ms  264/10509 (2.5%)
[  3] 17.0-18.0 sec  14.2 MBytes   119 Mbits/sec   0.088 ms  211/10346 (2%)
[  3] 18.0-19.0 sec  14.3 MBytes   120 Mbits/sec   0.106 ms  295/10497 (2.8%)
[  3] 19.0-20.0 sec  14.1 MBytes   118 Mbits/sec   0.042 ms  233/10277 (2.3%)
[  3] 20.0-21.0 sec  14.0 MBytes   117 Mbits/sec   0.420 ms  332/10288 (3.2%)
```

Figure 3-8: Iperf server side



type	rate	throughput	ewma prob	this prob	retry	this succ/attempt	success	attempts	
HT20/LGI	MCS0	6.7	100.0	100.0	1	0(0)	1	1	
HT20/LGI	MCS1	13.2	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS2	19.8	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS3	26.3	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS4	39.2	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS5	51.2	100.0	100.0	5	0(0)	1	1	
HT20/LGI	MCS6	57.2	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS7	64.5	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS8	13.2	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS9	26.3	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS10	39.2	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS11	51.2	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS12	76.4	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS13	97.1	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS14	110.1	100.0	100.0	6	0(0)	4	4	
HT20/LGI	MCS15	120.6	100.0	100.0	3	0(0)	8381	8381	P
HT20/LGI	MCS16	19.8	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS17	39.2	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS18	57.2	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS19	76.4	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS20	110.1	100.0	100.0	0	0(0)	1	1	
HT20/LGI	MCS21	141.2	99.9	100.0	4	5(5)	164963	166097	t
HT20/LGI	MCS22	147.8	92.8	95.7	4	564(589)	837715	934982	T
HT20/LGI	MCS23	20.7	12.2	0.0	0	0(5)	1239	8983	

Total packet count:: ideal 1002646 lookaround 10045
Average A-MPDU length: 16.6

Figure 3-9: Rate table while running iperf command

In order to calculate the PDR, a script in the BASH (Bourne Again SHell) command language, was designed (**Figure 3-10**). This script takes as input two arguments:

1. Time interval (in seconds) during which the script is executed
2. MCS for which the PDR and the aggregation level are computed

The output of the script is:

1. The PDR percentage
2. The number of packets used to form an AMPDU (aggregation level)

```
root@node092:~# bash pdr.sh 10 MCS22
PDR:
90.6594
AMPDU length:
16.6889
```

Figure 3-10: Script for calculating the PDR and the AMPDU length

For the throughput calculation the output of the iperf command is used. As shown in **Figure 3-11**, iperf command ran for 166 seconds, totally sending 2.33 GBytes of data with a rate of 120 Mbits/sec.

Figure 3-11: Retrieving throughput using iperf's output

3.5 Results

In the following section, throughput, PDR and jitter of the default driver for every scenario and every topology are presented.

3.5.1 High SNR Scenarios

Specifically, for the high SNR scenarios, network topologies with good link quality were chosen. For all these topologies throughput, PDR and jitter were measured for different MTU sizes (1500, 762, 516, 394) (**Figure 3-12**). It is obvious that when the quality of the channel is good, higher throughput is achieved for higher MTU values (**Figure 3-13**). For example, for a scenario where one antenna is used, for MTU 1500, MCS 7 is chosen and the achieved throughput is 58,1 Mbps, whilst for the MTU 516 for the same MCS the throuput is decreased to 51 Mbps. Although the fact that longer frames result to higher error rate, in high SNR the condition of the link is good enough, so most of the packet transmissions are successful (i.e. PDR is increased), retransmissions are limited and overhead is decreased. So it is desirable to maximize the number of bytes sent with the least overhead.

Node 085 -> Node 087			
Txpower: 9			
MTU 1500			
Rate	Throughput	PDR	Jitter
MCS 6	52,3	100	0,450495
MCS 7	58,1	100	0,403533
MTU 762			
Rate	Throughput	PDR	Jitter
MCS 6	49,6	100	0,252905
MCS 7	54,7	100	0,229097
MTU 516			
Rate	Throughput	PDR	Jitter
MCS 6	46,2	100	0,180954
MCS 7	51	100	0,163825
MTU 394			
Rate	Throughput	PDR	Jitter
MCS 6	43,4	99,9756	0,144341
MCS 7	47,8	99,9778	0,130984

Figure 3-12: Throughput, jitter and PDR for High SNR scenarios for varying MTU size

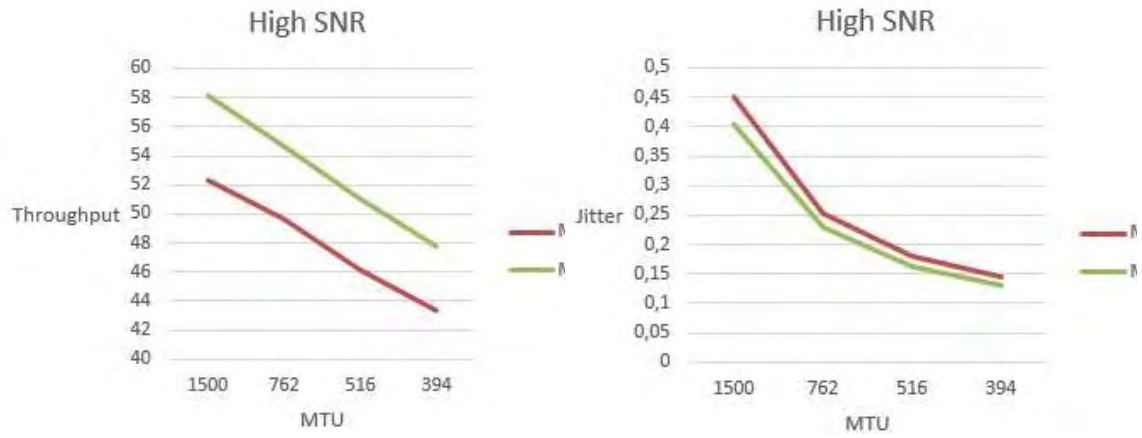


Figure 3-13: Graphs for throughput and jitter in high SNR scenarios

3.5.2 Low SNR Scenarios

The low SNR scenarios experiments took place in the EKETA's office testbed. For the low SNR scenarios, the MTU was set in such a way that one packet was divided in an exact number of smaller parts in order to reduce overhead. The packet size used in the iperf command was standard (1500 bytes), so in order to divide them in smaller packets the MTU was chosen taking into consideration the size of the IP header and the AMPDU overhead. Indicatively, with MTU 762 the packet was divided in two packets. So these packets form a larger packet with total length 1524 bytes, where 1500 bytes are the payload, 20 bytes the IP header and 4 bytes the AMPDU overhead. **Figure 3-14** shows the results of the default driver for the fixed MCS 5. In this graph, the correlation between MTU, throughput, PDR and jittter when the channel quality is low, is obvious. As long as the MTU is decreased, throughput is increased whereas jitter is decreased (**Figure 3-15**, **Figure 3-16**). This happens due to the fact that the link quality is not good enough, so it is more difficult for larger packets to be received. By reducing the size of the packet, the PDR is increased, retransmission are reduced and as a consequence throughput is enhanced.

MTU 1500

Category	MCS	PDR	Throughput (Mbits/sec)	Aggregation Level	Jitter	UDP loss (%)
T	5	1,86	0,35	7,6	117,99	78

MTU 762

Category	MCS	PDR	Throughput (Mbits/sec)	Aggregation Level	Jitter	UDP loss
T	5	8,66	2,42	14,55	81,11	27

MTU 516

Category	MCS	PDR	Throughput (Mbits/sec)	Aggregation Level	Jitter	UDP loss
T	5	38,22	13,9	13,97	15,97	12

MTU 394

Category	MCS	PDR	Throughput (Mbits/sec)	Aggregation Level	Jitter	UDP loss
T	5	43,91	14,2	13,64	3,29	11

MTU 320

Category	MCS	PDR	Throughput (Mbits/sec)	Aggregation Level	Jitter	UDP loss
T	5	84,68	26,4	21,51	1,01	10

MTU 274

Category	MCS	PDR	Throughput (Mbits/sec)	Aggregation Level	Jitter	UDP loss
T	5	88,61	26,1	20,09	1,04	12

Figure 3-14: Low SNR scenario with the rate fixed to MCS 5

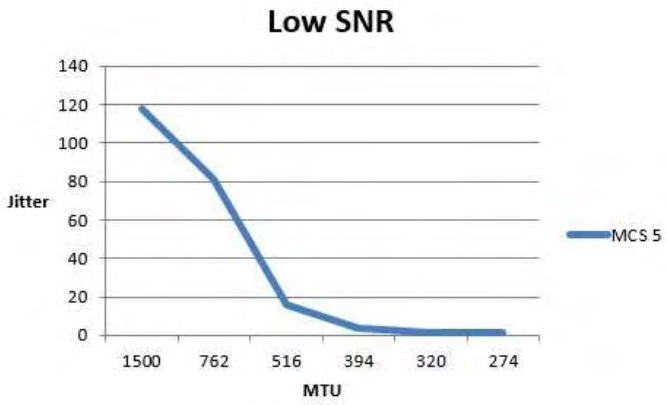


Figure 3-15: Graph for jitter for Low SNR scenarios with a fixed rate (MCS 5)

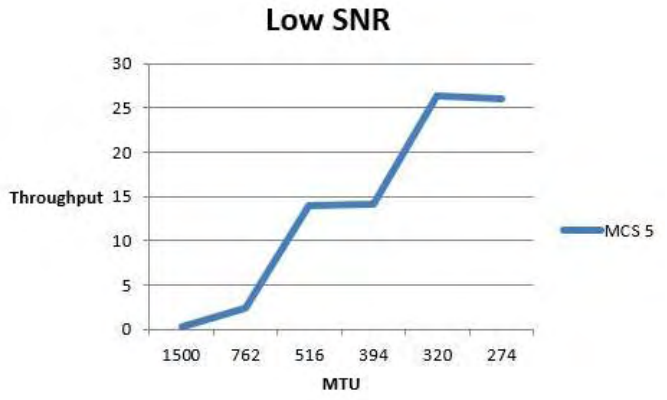


Figure 3-16: Graph for throughput for Low SNR scenarios with a fixed rate (MCS 5)

Figure 3-17 presents the results for the same topology but with the rate fixed to MCS 4. In this scenario, the results are not as clear as in the previous one but still there is still an improvement in throughput (**Figure 3-18**), if MTU 762 is chosen instead of the default MTU 1500. The reason for which the other MTUs doesn't show improvement is that when decreasing the packet size more overhead is introduced and this cannot be covered from the throughput gain due to PDR increase. For jitter the best value is achieved for the MTU 320 (**Figure 3-19**).

In both experiments, jitter and UDP loss are significantly decreased for smaller MTU. Especially, in the first experiment the gain for jitter is almost 100% and for UDP loss is 84%.

EKETA Office Testbed						
MTU 1500						
Category	MCS	PDR	Throughput (Mbits/sec)	Aggregation Level	Jitter	UDP loss (%)
T	4	80,14	27,1	11,56	2,77	3
MTU 762						
Category	MCS	PDR	Throughput (Mbits/sec)	Aggregation Level	Jitter	UDP loss
T	4	87,83	28,5	18,91	1,31	5
MTU 516						
Category	MCS	PDR	Throughput (Mbits/sec)	Aggregation Level	Jitter	UDP loss
T	4	93,88	28	27,08	1,36	5
MTU 394						
Category	MCS	PDR	Throughput (Mbits/sec)	Aggregation Level	Jitter	UDP loss
T	4	95,59	26,2	25,28	1,04	5
MTU 320						
Category	MCS	PDR	Throughput (Mbits/sec)	Aggregation Level	Jitter	UDP loss
T	4	98,48	26,8	30,49	0,87	7
MTU 274						
Category	MCS	PDR	Throughput (Mbits/sec)	Aggregation Level	Jitter	UDP loss
T	4	98,65	26,5	31,16	1,01	4

Figure 3-17: Low SNR scenario with a fixed rate (MCS 4)

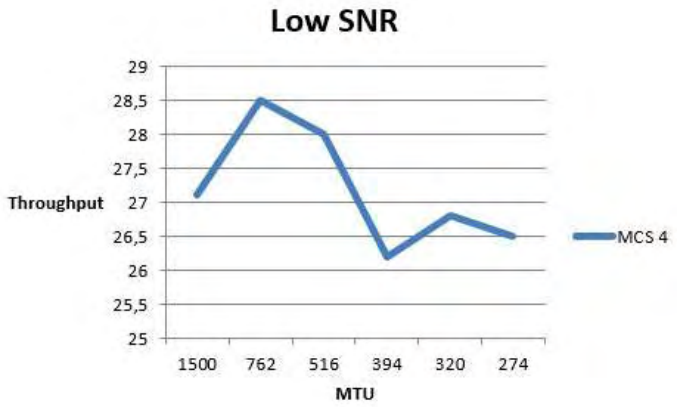


Figure 3-18: Graph for throughput for Low SNR scenarios with a fixed rate (MCS 4)

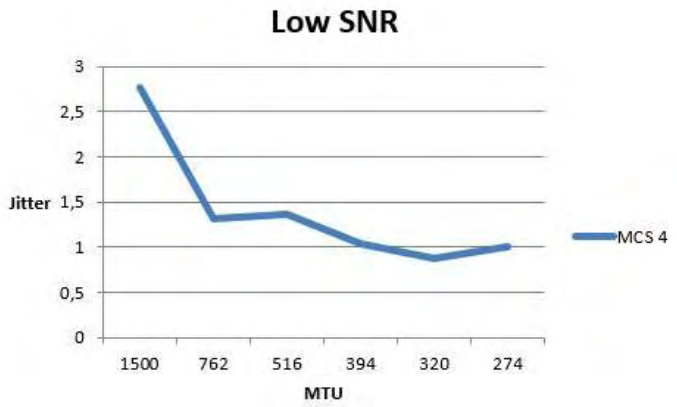


Figure 3-19: Graph for jitter for Low SNR scenarios with a fixed rate (MCS 4)

In order to verify the above observations, the experiment was executed in different low SNR topologies. From **Figure 3-20** is clear that the best throughput is not achieved for the default MTU (i.e. 1500) but for MTU 516.

Node 050 -> Node 085			
Txpower: 4			
MTU 1500			
Rate	Throughput	PDR	Jitter
MCS 3	23	100	1,02744
MCS 4	30,8	89,5553	0,765506
MCS 5	0,884	0,107948	27,8354
MTU 762			
Rate	Throughput	PDR	Jitter
MCS 3	22,1	100	0,533362
MCS 4	31,2	93,8307	0,377002
MCS 5	0,768	1,62904	15,5656
MTU 516			
Rate	Throughput	PDR	Jitter
MCS 2	15,8	99,9527	0,498301
MCS 3	21,2	100	0,369467
MCS 4	31,3	98,2743	0,263841
MCS 5	0,963	6,80918	8,52824
MTU 394			
Rate	Throughput	PDR	Jitter
MCS 3	20,4	99,9531	0,288919
MCS 4	29	96,5179	0,210861
MCS 5	1,14	8,07244	4,9055

Node 050 -> Node 085			
Txpower: 3			
MTU 1500			
Rate	Throughput	PDR	Jitter
MCS 2	17,1	100	1,38214
MCS 3	22,9	99,8636	1,02802
MCS 4	20,5	58,6031	1,12464
MTU 762			
Rate	Throughput	PDR	Jitter
MCS 2	16,5	100	0,71752
MCS 3	22,1	100	0,533476
MCS 4	24,8	76,0623	0,473565
MTU 516			
Rate	Throughput	PDR	Jitter
MCS 2	15,8	100	0,498258
MCS 3	21,2	100	0,370272
MCS 4	28,7	91,8081	0,280075
MCS 5	0,72	0,216124	11,2688
MTU 394			
Rate	Throughput	PDR	Jitter
MCS 2	15,2	100	0,388286
MCS 3	20,4	100	0,288883
MCS 4	25,3	84,5462	0,233477

Figure 3-20: Throughput, jiiter and PDR for another low SNR topology for different rates

The improvement in jitter and UDP loss is very important for video and voice applications. More specifically, packets are sent in a continuous stream but due to network congestion the delay between each packet can vary. The router that receives a Real Time Protocol (RTP) audio stream for Voice over IP (VoIP) has a mechanism to buffer the packets of the stream, before it will reproduce them as a steady stream. If the jitter is large enough to delay the reception of the packets, the quality of the sound will deteriorate (i.e. there will be sound interruptions). In addition, better quality is achieved when the UDP loss is as small as possible.

3.5.3 Hidden Terminal Scenarios

One of the most common problems in wireless networking is the hidden terminal problem. Nodes in a wireless network are called hidden when there are out of range of other nodes. As it is shown in **Figure 3-21** AP (node B) is visible from node A and from node C, however, node A is outside of the transmission range of node C and vice versa.

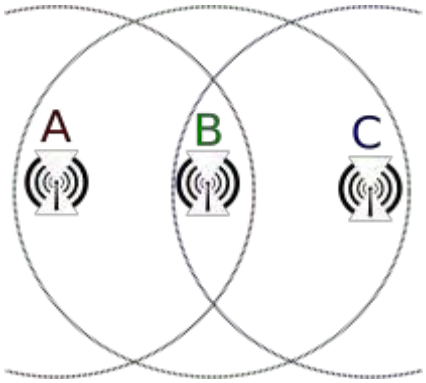


Figure 3-21: Hidden Terminal Problem

The problem occurs when node A and node C start to send packets simultaneously. Since node A is out of range for the node C, CSMA/CA doesn't work properly, so collissions happen and packets are lost. In order to reduce the frame collision, Request to Send / Clear to Send (RTS/CTS) mechanism is used. In the experiments that are described below, the RTS/CTS mechanism is disabled.

In **Figure 3-22** - **Figure 3-26** are presented the results of two runs from the hidden terminal experiment where the hidden link has fixed rate and constant traffic (5 Mbps). Throughput, jitter and PDR are calculated for STA for fixed MCS. In both cases, the results showed that smaller MTUs than the default MTU, improve throughput, PDR and jitter. More specific, for the first scenario where the hidden link uses MCS 1, the best throughput is achieved for the MTU 516 (MCS 4 – 23,5 Mbps). PDR and jitter acquire their maximum values for the MTU 394. For the second scenario wher the hidden link operates at MCS 6, best throughput, PDR and jitter are achieved for MTU 394. This behaviour can be explained from the fact that as traffic of the hidden link is constant and delay is big enough due to the small MCS, the collision - free time slots between packet

transmissions are not sufficient in order to let the STA transmit a packet in a higher MTU. On the other hand, if the smallest MTU is used, the gain in throughput will be minimized due to the overhead. When a higher rate is used in the hidden link the transmission time for the packet is decreased, so there are more collision – free timeslots resulting to increased throughput and PDR.

MCS 1 - Traffic 5 Mbps			
MTU 1500			
Rate	Throughput	PDR	Jitter
MCS 3	15,7	67,8	1,29
MCS 4	22,9	67,4	0,93
MCS 5	11,3	23,5	1,76
MTU 762			
Rate	Throughput	PDR	Jitter
MCS 3	16,9	74,9	0,65
MCS 4	23,1	70,1	0,5
MCS 5	12,6	28,7	0,82
MTU 516			
Rate	Throughput	PDR	Jitter
MCS 3	16,9	79	0,44
MCS 4	23,5	76,4	0,31
MCS 5	14	34,3	0,51
MTU 394			
Rate	Throughput	PDR	Jitter
MCS 3	16,6	82	0,33
MCS 4	22,9	78,5	0,23
MCS 5	16,5	41,8	0,32

MCS 6 - Traffic 5 Mbps			
MTU 1500			
Rate	Throughput	PDR	Jitter
MCS 3	17,1	74	1,17
MCS 4	28	80	0,76
MCS 5	30,3	65,6	0,65
MTU 762			
Rate	Throughput	PDR	Jitter
MCS 3	18,5	83,8	0,59
MCS 4	28,8	85,8	0,38
MCS 5	30,1	70	0,34
MTU 516			
Rate	Throughput	PDR	Jitter
MCS 3	17,7	83	0,41
MCS 4	27,6	86,1	0,26
MCS 5	32,5	80,11	0,21
MTU 394			
Rate	Throughput	PDR	Jitter
MCS 3	17,3	85,3	0,32
MCS 4	26,5	89,3	0,21
MCS 5	34,2	87,4	0,16

Figure 3-22: Throughput, jitter and PDR for hidden terminal scenarios

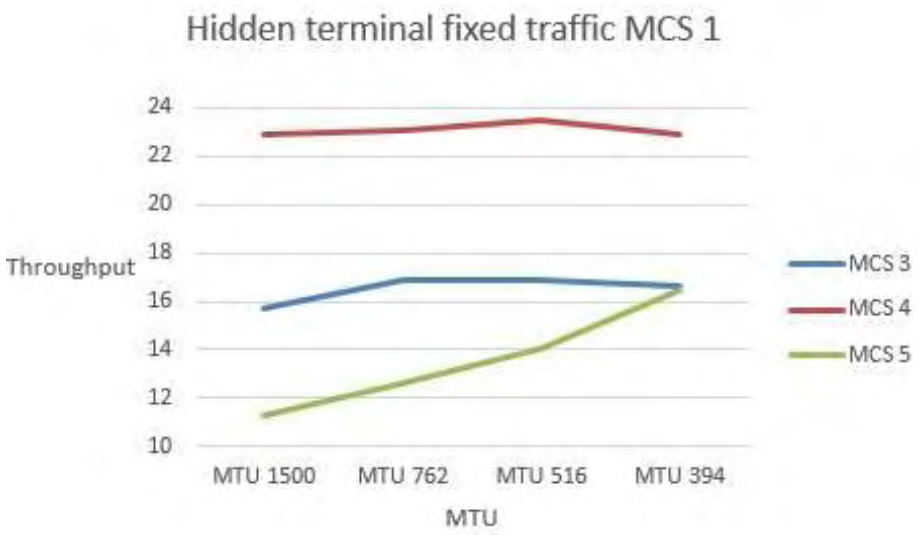


Figure 3-23: Throughput improvement for different MTU in hidden terminal scenario where the hidden link has constant traffic (5 Mbps) and fixed rate (MCS 1)

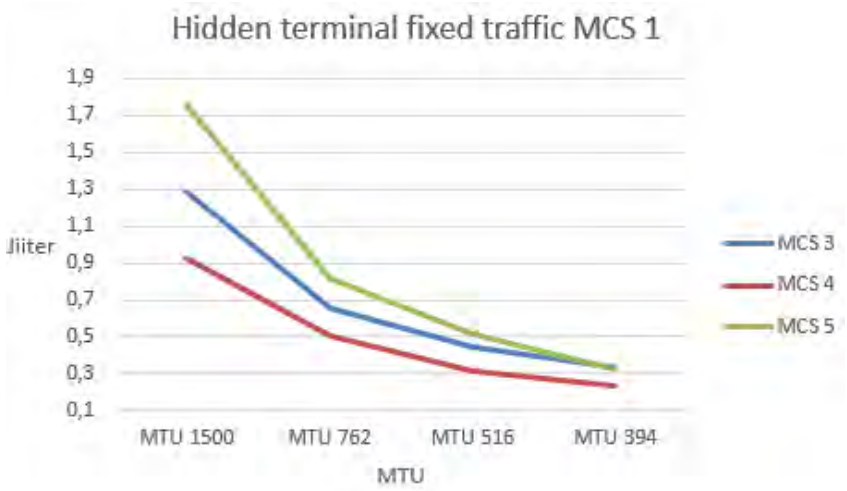


Figure 3-24: Jitter improvement for different MTU in hidden terminal scenario where the hidden link has constant traffic (5 Mbps) and fixed rate (MCS 1)

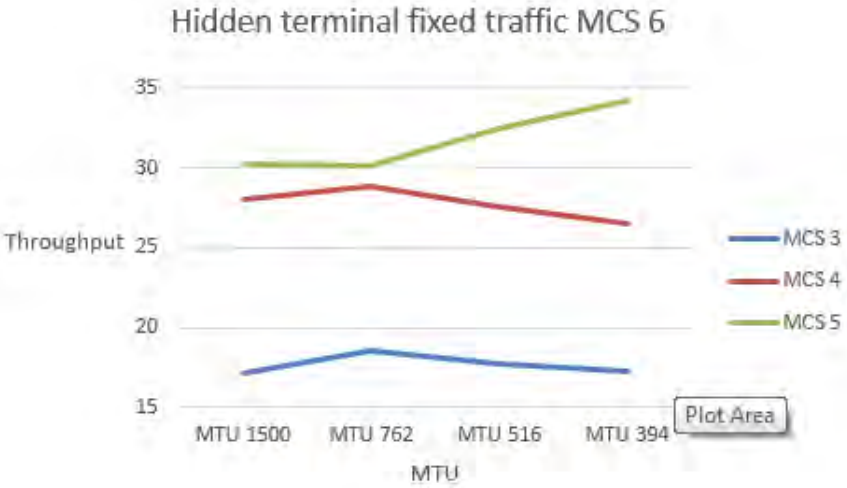


Figure 3-25: Throughput improvement for different MTU in hidden terminal scenario where the hidden link has constant traffic and fixed rate (MCS 6)

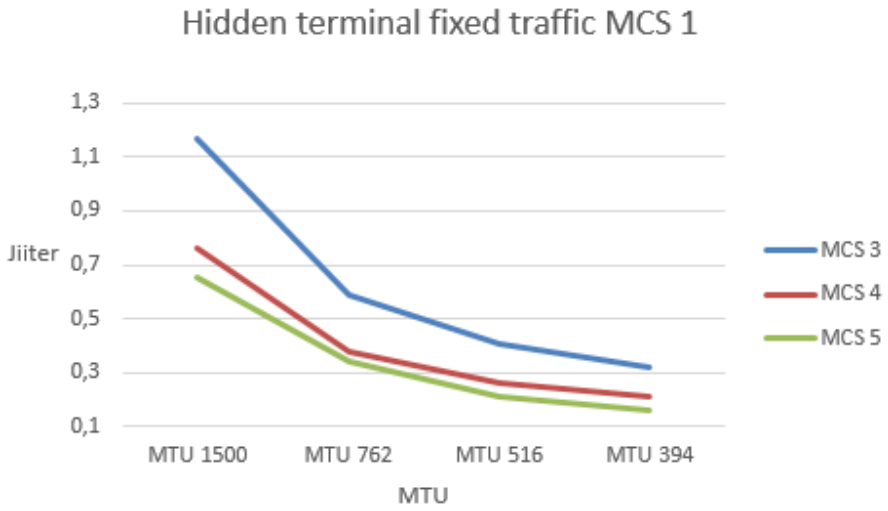


Figure 3-26: Throughput improvement for different MTU in hidden terminal scenario where the hidden link has constant traffic and fixed rate (MCS 6)

3.6 Implementation

Minstrel's rate adaptation algorithm was modified in order to automatically sample rates for different MTU sizes. In the default implementation the driver has a number of groups that are called *minstrel_mcs_groups*. These groups are used to define the MCS in which the data will be transmitted and furthermore to store information such as throughput and successfuly transmitted packets for every rate. They are categorized based on the number of streams that are used from the device, the guard interval (SGI or LGI) and the width of the channel (HT40 or HT20). For example, the MCS group (3, 1, 1) has 3 streams and uses SGI and HT40. These groups are also used for the probing mechanism. For every packet used for probing (10% of the transmitted packets), a group and an MCS index (0-7) are chosen. So the first packet of the ones used for probing, will be sent for the MCS group (1, 1, 1) and for the MCS index 0, the second one for the same MCS group but for the next MCS index (i.e. 1) and so on. However, there are some criteria based on which some rates are sampled occasionaly. For example if one of the chosen rate is a lower rate it won't be sampled as much as a higher rate. In **Figure 3-27** there is an abstract presentation of the packet flow and in **Figure 3-28** is the sampling process of the default driver.

In the new implementation, the main concept was to modify these groups in order to take also into consideration the MTU size (which affects the aggregation level). An example of the form of the new MCS groups is (3, 1, 1, 0). This group has 3 streams, uses SGI, has a channel bandwidth of 40 MHz and the MTU is equal to 1500 bytes. Now, the driver has the capability to change the MTU, probe different rates for different MTUs and therefore for different aggregation levels and store the results for different MTUs in the rate table as shown in the figure **Figure 3-30:**. Also, in order to have a more accurate value for throughput, MTU size was taken into consideration in the throughput calculation. In **Figure 3-29** is depicted the difference in sampling mechanism for the new implementation. By adding MTU size in the group formation, more options are available for choosing the more optimal rates for the existing link quality.

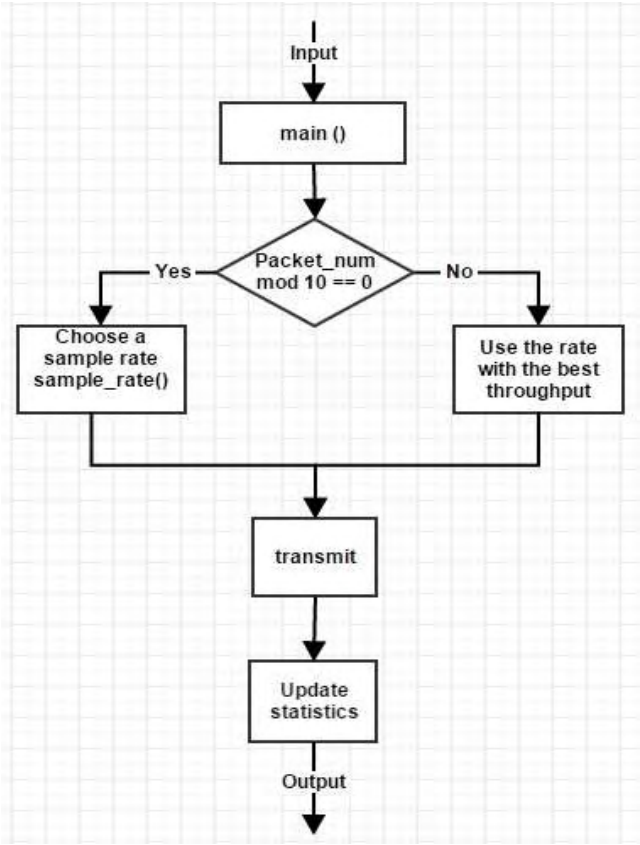


Figure 3-27: Packet processing

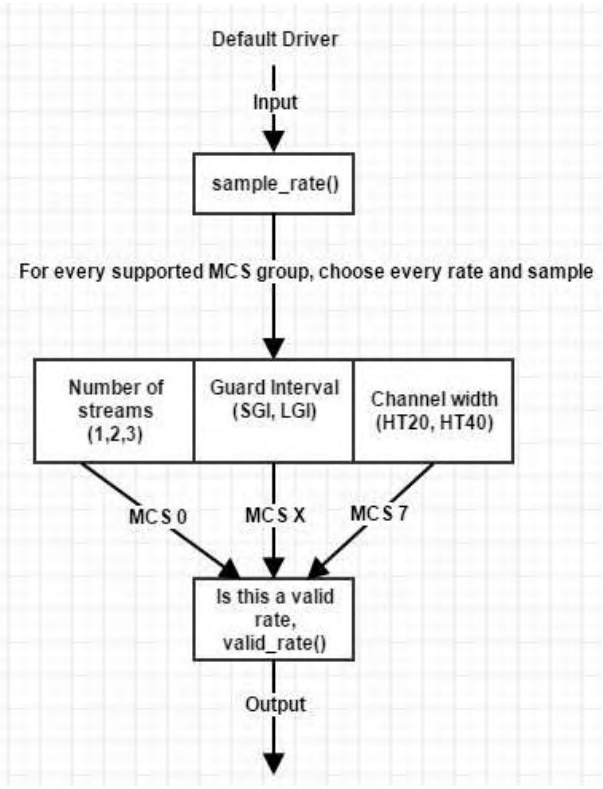


Figure 3-28: Sampling procedure

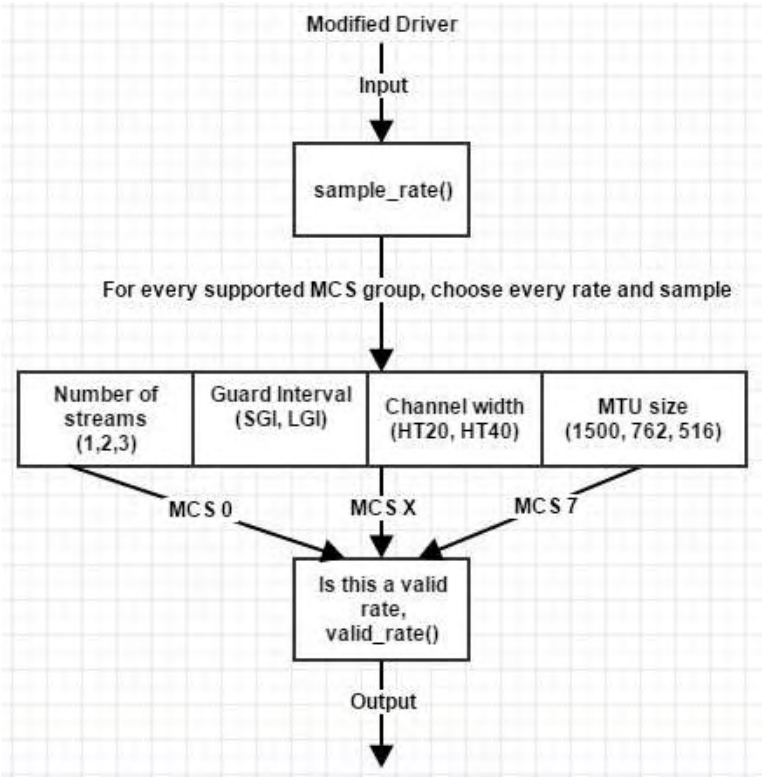
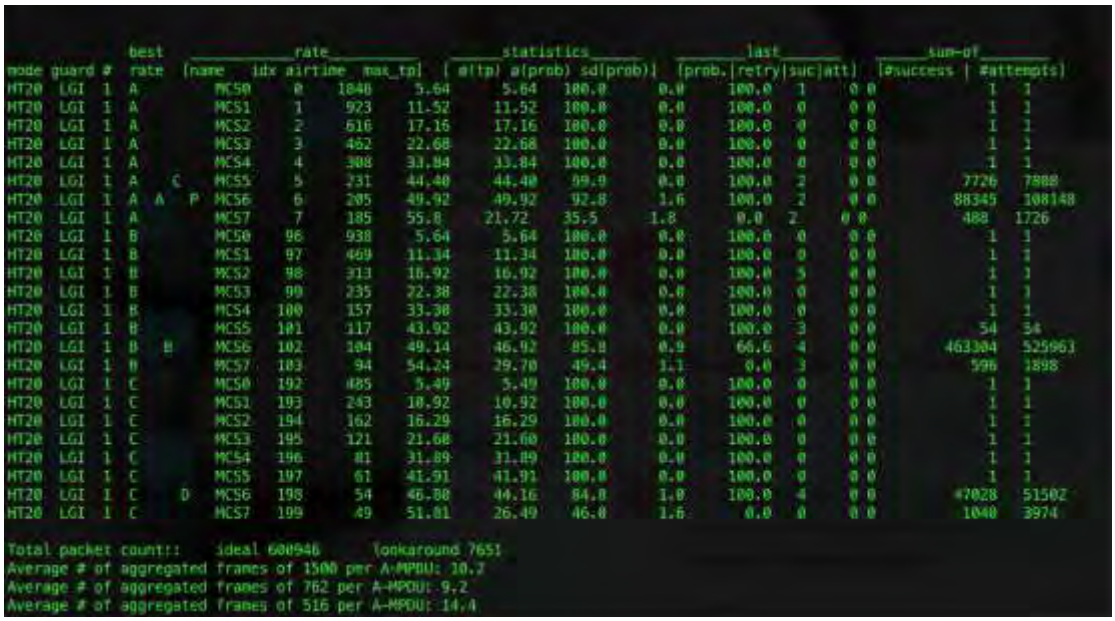


Figure 3-29: Sampling procedure of the modified driver

Finally, in **Figure 3-30** is the new rate table of the modified driver. In the new rate table three occurences of the MCS 0-7 exist, due to the three different MTU sizes.



	best		rate			statistics			last			sum-of	
mode guard #	rate	(name	idx	airtime	max_tp]	[a(tp)	a(prob)	sd(prob)]	(prob.	retry	suc	att]	(#success | #attempts)
HT20 LGI 1	A	MCS0	0	1046	5.64	5.64	100.0	0.0	100.0	1	0	0	1	1
HT20 LGI 1	A	MCS1	1	923	11.52	11.52	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	A	MCS2	2	616	17.16	17.16	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	A	MCS3	3	462	22.68	22.68	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	A	MCS4	4	308	33.84	33.84	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	A C	MCS5	5	231	44.40	44.40	99.9	0.0	100.0	2	0	0	7726	7808
HT20 LGI 1	A A P	MCS6	6	205	49.92	49.92	92.8	1.6	100.0	2	0	0	88345	108148
HT20 LGI 1	A	MCS7	7	185	55.8	21.72	35.5	1.8	0.0	2	0	0	488	1726
HT20 LGI 1	B	MCS0	96	938	5.64	5.64	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	B	MCS1	97	469	11.34	11.34	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	B	MCS2	98	313	16.92	16.92	100.0	0.0	100.0	5	0	0	1	1
HT20 LGI 1	B	MCS3	99	235	22.38	22.38	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	B	MCS4	100	157	33.30	33.30	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	B	MCS5	101	117	43.92	43.92	100.0	0.0	100.0	3	0	0	54	54
HT20 LGI 1	B B	MCS6	102	104	49.14	46.92	85.8	0.9	66.6	4	0	0	463304	525963
HT20 LGI 1	B	MCS7	103	94	54.24	29.70	49.4	1.1	0.0	3	0	0	596	1898
HT20 LGI 1	C	MCS0	192	485	5.49	5.49	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	C	MCS1	193	243	10.92	10.92	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	C	MCS2	194	162	16.29	16.29	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	C	MCS3	195	121	21.60	21.60	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	C	MCS4	196	81	31.89	31.89	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	C	MCS5	197	61	41.91	41.91	100.0	0.0	100.0	0	0	0	1	1
HT20 LGI 1	C D	MCS6	198	54	46.80	44.16	84.0	1.0	100.0	4	0	0	47028	51502
HT20 LGI 1	C	MCS7	199	49	51.81	26.49	46.0	1.6	0.0	0	0	0	1040	3974

Total packet count:: ideal 600946 lonkaround 7651
Average # of aggregated frames of 1500 per A-MPDU: 10.2
Average # of aggregated frames of 762 per A-MPDU: 9.2
Average # of aggregated frames of 516 per A-MPDU: 14.4

Figure 3-30: rate table (rc_stats) of the modified driver

3.7 Driver Ath9k

Ath9k is a Linux kernel driver for all Atheros IEEE 802.11n PCI/PCI-Express and AHB WLAN based chipsets introduced at Linux 2.6.27. In Ath9k, Minstrel can be be used as a rate control algorithm in order to boost the performance. For this thesis backports-3.17.1-1 and 4.2.6.1 versions were used.

3.8 Comparison between the default and the modified driver

In order to show the improvement of throughput using the modified driver, similar experiments as the ones described in the previous section were done. Indoor RF Isolated testbed was used. Specifically, a pair of nodes was chosen, based on the link quality (**Figure 3-31**).

Figure 3-31: Example of wlan0 configuration

3.8.1 Low SNR Scenarios

In order to deteriorate the link quality for the low SNR scenarios, transmission power was decreased to 5 dBm and one antenna with attenuator was used. In these tests rate control algorithm automatically chooses the transmission rate. Two nodes were used (node050 as the AP and node091 as the STA). In every experiment different number of antennas (on both AP and STA) were used. For the whole range of transmission power (txpower) in every setup, throughput and jitter were measured. By modifying the number of antennas and the transmission power for the same nodes, different network topologies are simulated. Throughput was measured from the iperf output. In order to calculate jitter a monitor interface was set up in AP in order to capture the packets and with the usage of a BASH script, jitter was calculated. The steps of the experiment were:

1. Set up node050 as the AP.
2. Connect node091 (STA) to the AP.
3. Set up a monitor interface in the AP in order to capture the traffic.
4. Send traffic from the STA to the AP for 30 seconds so that algorithm can converge to the correct rate.

5. Send traffic from the STA to the AP for 30 seconds and capture the traffic.
6. Measure throughput from the output of the iperf command and calculate the jitter from the capture file.
7. Repeat the steps 4-6 for all the values of transmission power.

In **Figure 3-32** - **Figure 3-34**, throughput and jitter are depicted for the default and the modified driver when one antenna is used in both AP and STA.

Topology (node050 AP --> node091 Client) 1 Antenna			
Default Driver			
Txpower	Throughput iperf	Throughput	Jitter
0	15	12	1,79178
1	15	13,3	1,62396
2	17	13,5	1,60452
3	18	13,5	1,60954
4	21	15,9	1,37031
5	23	16,8	1,31299
6	23	17,5	1,25672
7	26	20,4	1,05613
8	29	22,4	0,965602
9	32	25,3	0,871404
Modified Driver			
Txpower	Throughput iperf	Throughput	Jitter
0	15	12,5	1,15821
1	17	13,3	1,37738
2	18	13,5	1,2979
3	18	13,8	1,42977
4	21	16,7	0,960766
5	23	18,4	0,938086
6	23	18,8	1,13148
7	26	20,9	0,551423
8	32	27,3	0,769722
9	33	27,4	0,625251

Figure 3-32: Throughput and jitter in low SNR topology, using one antenna

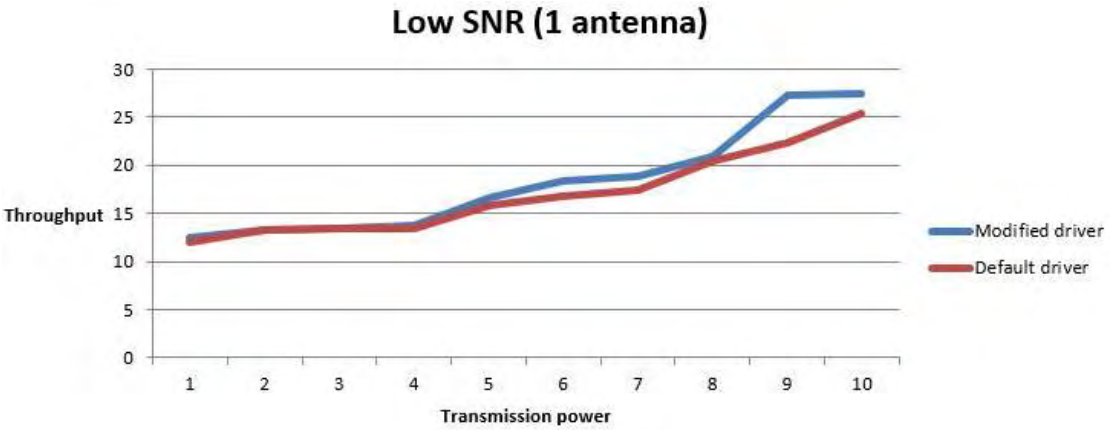


Figure 3-33: Throughput improvement with the modified driver in low SNR topology, using one antenna

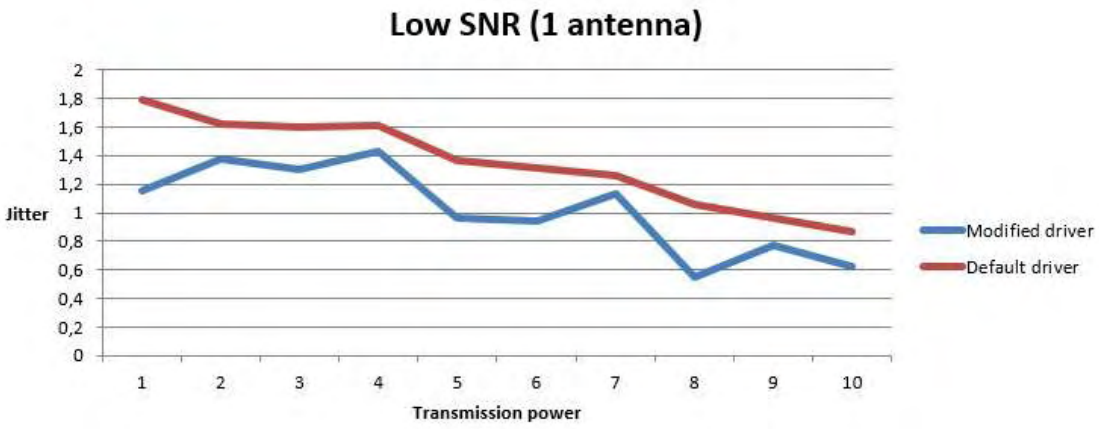


Figure 3-34: Jitter improvement with the modified driver in low SNR topology, using one antenna

From the results it is obvious that modified driver achieves better or equal throughput in all cases with an average throughput improvement of 7% and a max improvement of 22%. Moreover, in all cases jitter is decreased with an average improvement of 23%.

In **Figure 3-35** - **Figure 3-37**, throughput and jitter are presented for the default and the modified driver when two antennas are used in both AP and STA.

Topology (node050 AP --> node091 Client) 2 Antennas			
Default Driver			
Txpower	Throughput iperf	Throughput	Jitter
3	25	21,9	1,01289
4	26	21,9	1,01536
5	29	25,2	0,846235
6	35	31,1	0,705137
7	44	39,7	0,545812
8	46	42,3	0,514775
9	48	43,8	0,504297
10	48	42,2	0,521762
11	50	44,9	0,485542
12	65	59	0,36821
Modified Driver			
Txpower	Throughput iperf	Throughput	Jitter
3	25	21,9	0,988383
4	26	21,9	0,984571
5	30	26,7	0,414888
6	36	31,5	0,647555
7	45	40,5	0,52574
8	48	43,5	0,497785
9	49	44,1	0,494557
10	49	41,3	0,36482
11	50	45,5	0,200925
12	67	63,2	0,341589

Figure 3-35: Throughput and jitter in low SNR topology, using two antennas

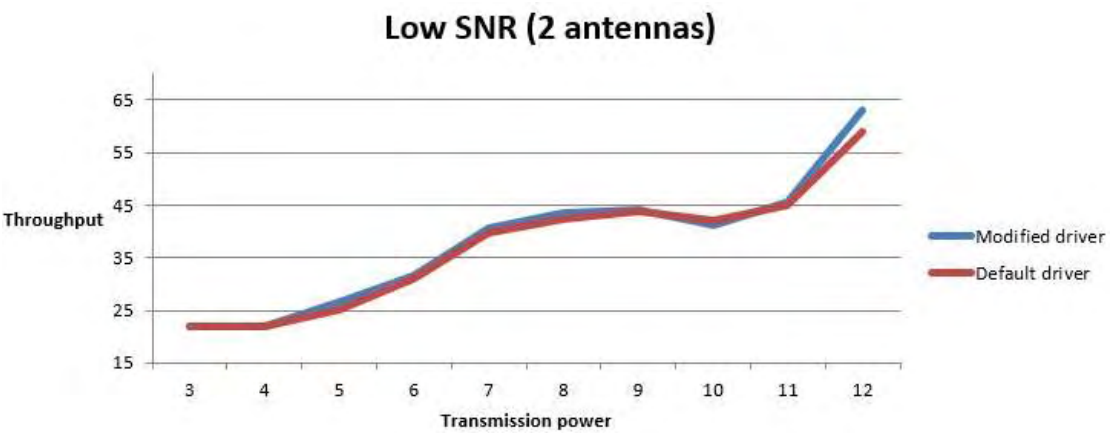


Figure 3-36: Throughput improvement with the modified driver in low SNR topology, using two antennas

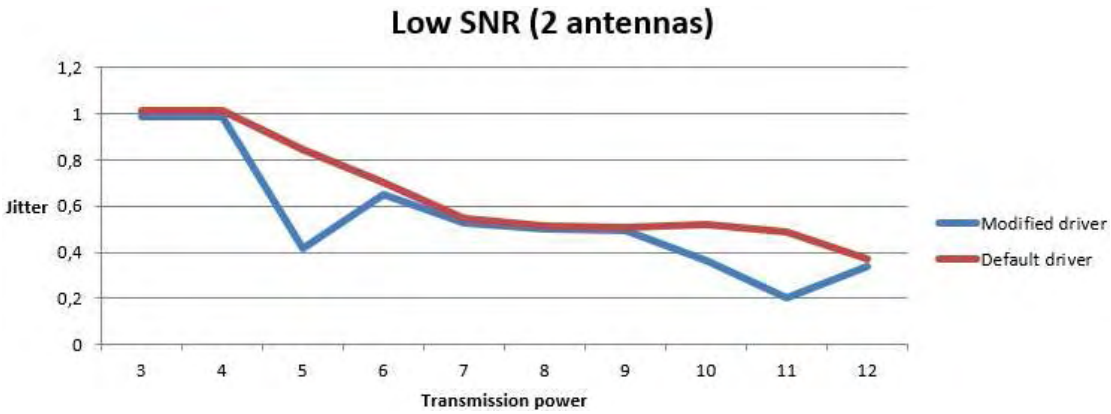


Figure 3-37: Jitter improvement with the modified driver in low SNR topology, using two antennas

In this experiment the throughput is better or equal to 90% of the cases with an average throughput improvement of 2,1% and a maximu improvement of 7%. In terms of jitter a 16% improvement was achieved for all cases.

In **Figure 3-38 - Figure 3-40**, the results of the experiments, when three antennas are used in both AP and STA, are shown.

Topology (node050 AP --> node091 Client) 3 Antennas			
Default Driver			
Txpower	Throughput iperf	Throughput	Jitter
5	50	46,8	0,463814
6	52	48,2	0,457427
7	53	48,2	0,457153
8	53	54,4	0,467404
9	64	59,8	0,361866
10	68	64,5	0,342061
11	69	64,6	0,343362
12	89	85,3	0,251796
13	98	94,6	0,239269
14	101	97	0,238479
Modified Driver			
Txpower	Throughput iperf	Throughput	Jitter
5	52	47,8	0,452743
6	53	48,3	0,448859
7	53	47,5	0,251754
8	65	62,2	0,344249
9	68	64,7	0,335787
10	69	65	0,334951
11	64	59,1	0,179545
12	93	87,6	0,246553
13	100	96,4	0,235097
14	101	96,9	0,234617

Figure 3-38: Throughput and jitter in low SNR topology, using three antennas

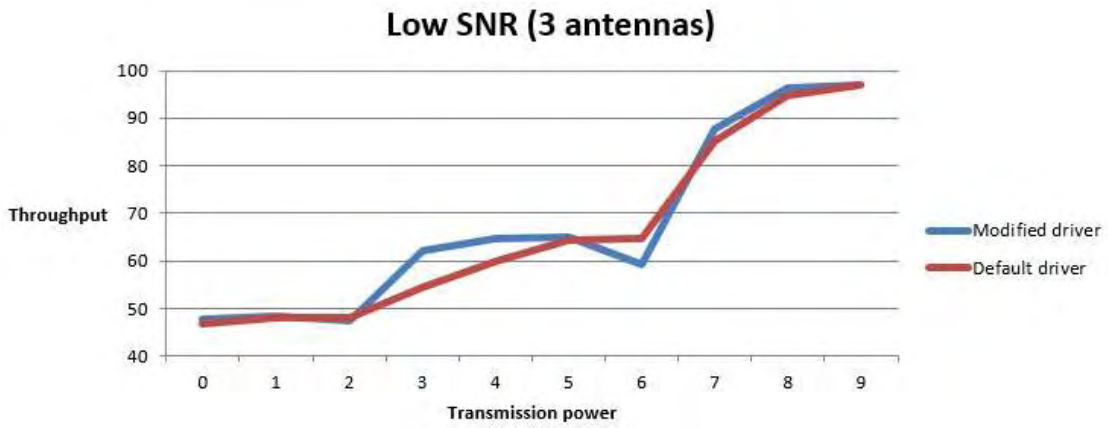


Figure 3-39: Throughput improvement with the modified driver in low SNR topology, using three antennas

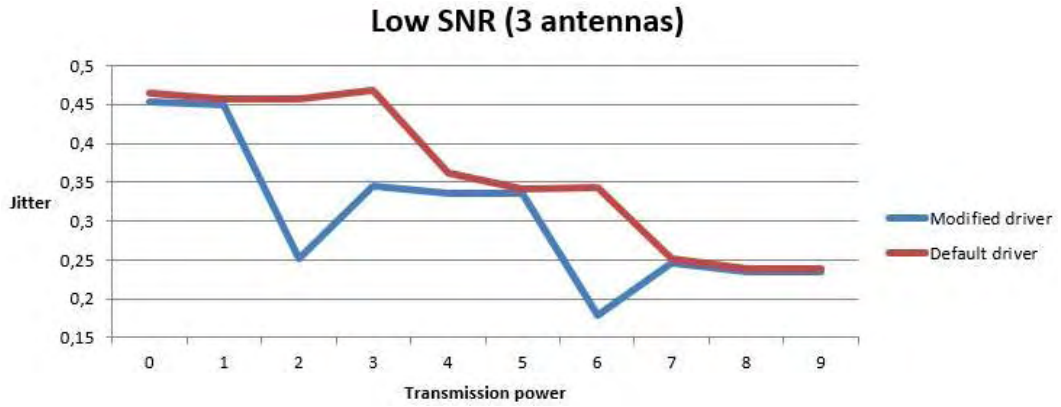


Figure 3-40: Jitter improvement with the modified driver in low SNR topology, using three antennas

Finally, in this topology the throughput is improved in 70% of the cases with an

average throughput improvement of 2,6% and maximum improvement of 14%. The gain in multi antennas topology is smaller because the default driver has a larger search space to choose a rate, so it can achieve a more precise to the receive sensitivity match. Again, jitter is enhaced in all cases with an average improvement of 15,5%.

3.8.2 High SNR Scenarios

For the high SNR, the experiments were similar to low SNR ones. The main difference was the better link quality. From the results as shown in **Figure 3-41** - **Figure 3-43** it is verified that throughput is enhanced in 70% of the cases with an average improvement of 3% and a maximum improvement of 16%. Jitter is enhanced in all scenarios with an average improvement of 20%.

Topology (node050 AP --> node085 Client) 1 Antenna			
Default Driver			
Txpower	Throughput iperf	Throughput	Jitter
0	25	20,7	1,04526
1	26	21,4	1,01597
2	26	21,3	1,01756
3	30	24,2	0,882112
4	34	29,7	0,728155
5	35	31,4	0,692117
6	36	32	0,67696
7	40	35,6	0,609067
8	45	41,5	0,525641
9	47	42,7	0,512853
10	48	42,7	0,513561
11	56	50,6	0,463805
Modified Driver			
Txpower	Throughput iperf	Throughput	Jitter
0	25	21,6	0,990206
1	27	22,2	0,973307
2	27	21,5	0,70284
3	32	28,1	0,447315
4	36	32	0,669606
5	38	33	0,652067
6	37	31	0,424338
7	43	38,8	0,446334
8	47	42,5	0,507613
9	47	39,7	0,343023
10	50	45,7	0,447974
11	54	49,2	0,444119

Figure 3-41: Throughput and jitter in high SNR topology, using one antenna

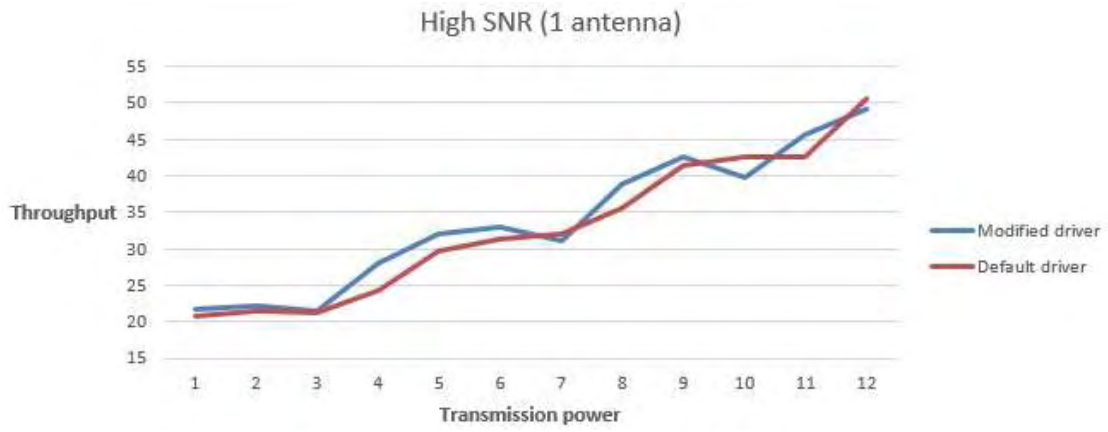


Figure 3-42: Throughput improvement with the modified driver in high SNR topology, using one antenna

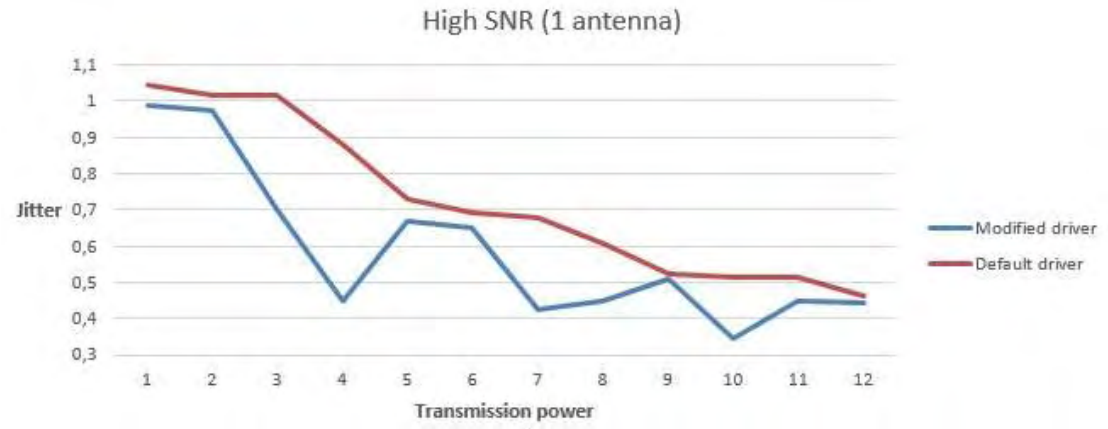


Figure 3-43: Jitter improvement with the modified driver in high SNR topology, using one antenna

3.8.3 Hidden Terminal Scenarios

For the hidden terminal scenario the following scenarios were implemented:

1. For fixed traffic (5 Mbps) and varying rates in the hidden link, throughput and jitter are measured for both drivers.
2. For fixed rate (MCS 5) and varying traffic rate in the hidden link, throughput and jitter are measured for both drivers.

In **Figure 3-44** - **Figure 3-49** are the results of the aforementioned experiments for hidden terminal problem. It is obvious that in both experiments the modified driver achieves better throughput and jitter in almost all rates and traffic. Especially, for the first scenario where the traffic of the hidden link is fixed, the highest throughput gain is achieved when the hidden link uses the highest rate (MCS 7). On the other hand, jitter is most improved when the hidden link uses

the MCS 6. In the second scenario where the rate of the hidden link is fixed, the highest throughput gain (75%) and jitter gain (67%) are achieved for the highest traffic (20 Mbps). This is a consequence of the smaller packet length that modified driver uses. When the hidden link uses low rate, it needs more airtime to transmit the data so collisions are most probable to happen. Also, more collisions can happen when the data rate is high so it is more possible to prevent collisions if the size of the packet is smaller.

Default Driver		
Fixed Traffic 5 Mbps		
Rate	Throughput	Jitter
MCS 0	19,6	0,957
MCS 1	19,9	1
MCS 2	21,3	0,94
MCS 3	21,7	0,89
MCS 4	23,8	0,81
MCS 5	24,8	0,78
MCS 6	25,6	0,74
MCS 7	24,2	0,79

Modified Driver		
Fixed Traffic 5 Mbps		
Rate	Throughput	Jitter
MCS 0	17,4	0,51
MCS 1	20,3	0,44
MCS 2	22,5	0,416
MCS 3	24	0,37
MCS 4	26,2	0,38
MCS 5	29,8	0,32
MCS 6	33,4	0,29
MCS 7	33,7	0,32

Results	
Gain (%)	
Throughput	Jitter
-11,2244898	46,70846395
2,010050251	56
5,633802817	55,74468085
10,59907834	58,42696629
10,08403361	53,08641975
20,16129032	58,974335897
30,46875	60,81081081
39,25619835	59,49367089

Figure 3-44: Gain for all supported rates in hidden terminal problem for fixed traffic

Default Driver		
Fixed MCS 5		
Traffic	Throughput	Jitter
1 Mbps	31,2	0,61
2 Mbps	30,6	0,63
5 Mbps	25,4	0,76
10 Mbps	18	0,97
20 Mbps	6,78	1,99

Modified Driver		
Fixed MCS 5		
Traffic	Throughput	Jitter
1 Mbps	36,5	0,32
2 Mbps	36,5	0,34
5 Mbps	30,3	0,31
10 Mbps	21,3	0,43
20 Mbps	11,9	0,65

Results	
Gain (%)	
Throughput	Jitter
16,98717949	47,54098361
19,28104575	46,03174603
19,29133858	59,21052632
18,33333333	55,67010309
75,51622419	67,33668342

Figure 3-45: Gain for the hidden terminal problem for fixed MCS and varying traffic

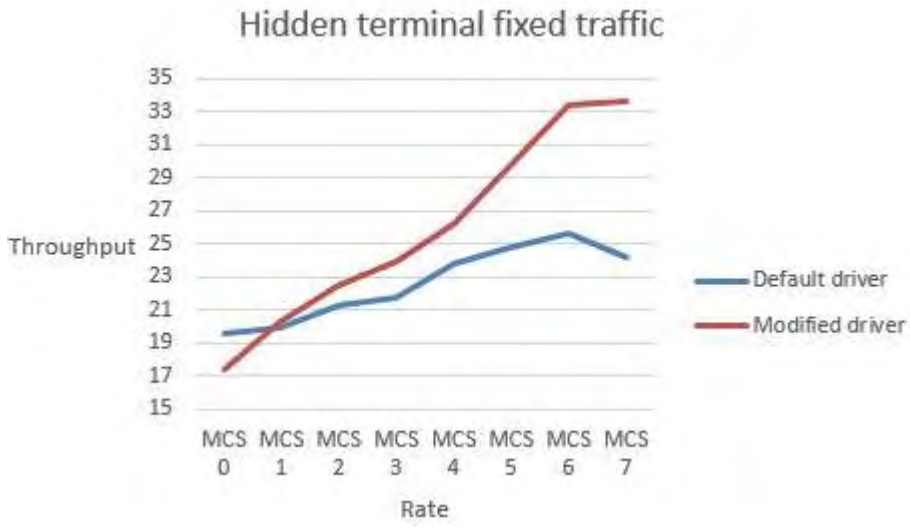


Figure 3-46: Throughput improvement of modified driver in hidden terminal scenario where the traffic of the hidden link is constant to 5 Mbps

Figure 3-47: Jiiter improvement of modified driver in hidden terminal scenario where the traffic of the hidden link is constant to 5 Mbps

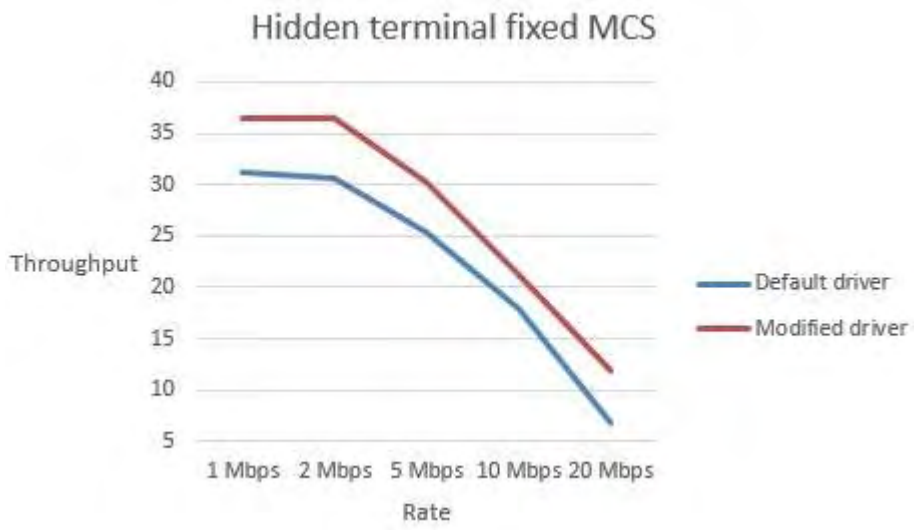


Figure 3-48: Throughput improvement of modified driver in hidden terminal scenario where the traffic of the hidden link is varying and the MCS is fixed (MCS 5)

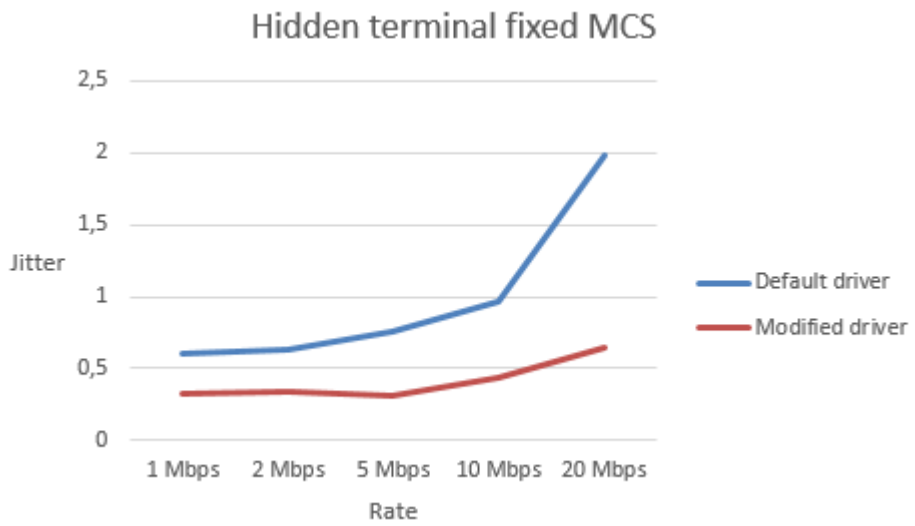


Figure 3-49: Jitter improvement of modified driver in hidden terminal scenario where the traffic of the hidden link is varying and the MCS is fixed (MCS 5)

4. SUMMARY

4.1 Conclusions

As wireless channel suffers from unstable conditions, rate adaptation is an integral part in order to exploit the channel. Rate adaptation's target is to determine the optimal rate for the constantly changing wireless channel conditions.

For this thesis minstrel's rate adaptation algorithm was used. The driver implemented in the scope of this thesis consists a significant study for the optimization of minstrel's rate adaptation algorithm. The enhanced rate adaptation algorithm provides better results in terms of throughput, jitter and PDR in conditions where the channel quality is not so optimal, like in medium SNR scenarios. Finally, it achieves great results for topologies that suffer from the hidden terminal problem, leading to a significant improvement of throughput.

4.2 Future Enhancements

There are a lot of possible future enhancements that are worth to implement and some aspects that should be analysed. One major drawback of driver ath9k is that restricts the usage of AMPDU aggregation simultaneously with fragmentation of frames in the MAC layer. In this implementation, IP fragmentation is used through MTU calibration and this results to introducing an extra IP overhead (20 bytes) for every MPDU. So, if fragmentation and aggregation can be used simultaneously, the algorithm would take more accurate decisions for which rate to use and better results will be achieved. Another important improvement is the implementation of a more accurate mechanism to calculate throughput. Due to the fact that the statistics depict the previous state of the channel, there are scenarios were the algorithm doesn't use the rate with the best throughput, so a better calibration of the calculated throughput to the actual throughput is needed.

Another field for improvement is the search space reduction for the sampling process. By adding MTU as a new dimension in the MCS groups, the number of possible rates that the algorithm can sample was increased. So, a more efficient way in the rate selection should be implemented.

Last but not least, it could be useful to evaluate if the modified rate algorithm is more energy efficient than the default one. Due to the fact that PDR is higher in the modified driver, less retransmissions will take place and less energy will be consumed

Finally, it can also be evaluated how video related metrics such as PSNR are affected from the modified driver.

REFERENCES

Dionysios Skordoulis, Qiang Ni, Hsiao-Hwa Chen, Adrian P. Stephens, Changwen Liu and Abbas Jamalipour. IEEE 802.11n MAC frame aggregation mechanisms for next-generation high-throughput wlans.

Thomas Paul and Tokunbo Ogunfunmi, 2008. Wireless LAN Comes of Age: Understanding the IEEE 802.11n Amendment.

Sourav Pal, Sumantra R. Kundu, Kalyan Basu and Sajal K. Das. IEEE 802.11 Rate Control Algorithms: Experimentation and Performance Evaluation in Infrastructure Mode.

Rolf Leutert, 2009. Inside 802.11n: Technical details about the new wlan standard.

Dong Xia, Jonathan Hart and Qiang Fu, 2013. Evaluation of the Minstrel Rate Adaptation Algorithm in IEEE 802.11g WLAN.

John C. Bicket, 2005. Bit-rate Selection in Wireless Networks.

Emilio Ancillotti, Raffaele Bruno and Marco Conti, 2008. Experimentation and Performance Evaluation of Rate Adaptation Algorithms in Wireless Mesh Networks.

Jangeun Jun, Pushkin Peddabachagari, Mihail Sichitiu. Theoretical Maximum Throughput of IEEE 802.11 and its Applications.

Nurul I. Sarkar. The Impact of Transmission Overheads on IEEE 802.11 Throughput: Analysis and Simulation.

Juniper Networks, 2011. Coverage or capacity- making the best use of 802.11n

Minstrel rate control algorithm, 2005.
"**https://wireless.wiki.kernel.org/en/developers/documentation/mac80211/ratecontrol/minstrel**"

Understanding jitter in packet voice networks (CISCO IOS platforms), 2006.
"**http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/18902-jitter-packet-voice.html**"

Bit-rate selection algorithms "http://madwifi-project.org/wiki/UserDocs/RateControl"

List of WLAN channels
"**https://en.wikipedia.org/wiki/List_of_WLAN_channels**"

IEEE 802.11n, 2009. "https://en.wikipedia.org/wiki/IEEE_802.11n-2009"